Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Iterative learning from positive data and counters

Timo Kötzing

Department 1: Algorithms and Complexity, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

ARTICLE INFO

Keywords:
Inductive inference
Function learning
Iterative learning
Transductive learning

ABSTRACT

We analyze iterative learning in the limit from positive data with the additional information provided by a *counter*. The simplest *type* of counter provides the current iteration number (counting up from 0 to infinity), which is known to improve learning power over plain iterative learning. We introduce five other (weaker) counter types, for example only providing some unbounded and non-decreasing sequence of numbers. Analyzing these types allows one to understand which *properties* of a counter learning can benefit from.

For the iterative setting, we completely characterize the relative power of the learning criteria corresponding to the counter types. In particular, for our types, the only properties improving learning power are *unboundedness* and *strict monotonicity*. Furthermore, we show that each of our types of counter improves learning power over weaker ones in *some* settings; to this end, we analyze *transductive* and *non-U-shaped* learning. Finally we show that, for iterative learning criteria with one of our types of counter, separations of learning criteria are necessarily witnessed by classes containing only infinite languages.

1. Introduction

We analyze the problem of algorithmically learning a description for a formal language (a computably enumerable subset of the set of natural numbers) when presented successively all and only the elements of that language. For example, a learner h might be presented with more and more even numbers. After each new number, h may output a description of a language as its conjecture. The learner h might decide to output a program for the set of all multiples of 4, as long as all numbers presented are divisible by 4. Later, when h sees an even number not divisible by 4, it might change this guess to a program for the set of all multiples of 2.

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 [12], gave a first, simple learning criterion, *TxtEx-learning*,[1] where a learner is *successful* iff, on every *text* for L (listing of all and only the elements of L) it eventually stops changing its conjectures, and its final conjecture is a correct description for the input sequence. Trivially, each single, describable language L has a suitable constant function as an Ex-learner (this learner constantly outputs a description for L). Thus, we are interested in characterizing for which *classes of languages* $\mathcal{L}$ is there a *single learner* h learning *each* member of $\mathcal{L}$. This framework is known as *language learning in the limit* and has been studied extensively, using a wide range of learning criteria similar to TxtEx-learning (see, for example, the textbook [13]).

In this paper we are concerned with a memory limited variant of TxtEx-learning, namely *iterative learning* [22,17] (**It**). While in TxtEx-learning a learner may arbitrarily access previously presented data points, in iterative learning the learner only sees its previous conjecture and the latest data point. It is well known that this setting allows one to learn strictly fewer classes of languages. Further work from the literature analyzed iterative learners with some additional resources, for

E-mail address: koetzing@mpi-inf.mpg.de.

[1] *Txt* stands for learning from a *text* of positive examples; *Ex* stands for *explanatory*.

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.09.023

example a *bounded example memory* [17]; "long term" *finite memory states* [11]; or *feedback learning*, i.e. the ability to ask for the membership of examples in previously seen data [17,5].

A different option for providing additional learning power for iterative learning was suggested in [10], where *iterative with counter learning* was introduced. In this setting, a learner, in each iteration, has access to its previous conjecture, the latest datum, and the current iteration number (counting up from 0 to infinity). Case and Moelius [10] show that this learning criterion is strictly more powerful than plain iterative learning, strictly less powerful than TxtEx-learning, and incomparable to *set-driven* learning [21]. In set-driven learning, the learner has access only to the (unordered) set of data seen so far, with duplicates removed. Consider now a learning criterion, where the learner has access to the set of data seen so far, just as in set-driven learning, but also to the current iteration number (just as in iterative with counter learning as introduced in [10]). It is easy to see that this learning criterion is equivalent to *partially set-driven* (or *rearrangement independent*) learning [20]; it is well known that partially set-driven learning is equivalent to TxtEx-learning.

The main aim of this paper is to discuss how and why such a counter improves learning power. In particular, we want to understand what properties of a counter can be used in a learning process to increase learning power. Is it the higher and higher counter values, which we can use to time-bound computations? Is it knowing the number of data items seen so far? Is it the complete enumeration of all natural numbers which we can use to divide up tasks into infinitely many subtasks to be executed at the corresponding counter value? We approach these questions by introducing different *counter types*, each modeling some of the possibly beneficial properties mentioned above. Formally, a counter type is a set of *counters*; a *counter* is a mapping from the set of natural numbers to itself. Instead of giving the learner the current iteration number, we will map this number with a counter drawn from the counter type under consideration.

We define the following counter types[2]:

(i) complete and ordered: $\mathrm{Id} = \{\mathrm{id}_{\mathbb{N}}\}$[3];
(ii) strictly monotone: $\vec{\mathrm{R}}^! = \{c \mid \forall i\colon\ c(i+1) > c(i)\}$;
(iii) monotone & unbounded: $\vec{\mathrm{R}} = \{c \mid \forall i\colon\ c(i+1) \geqslant c(i) \wedge \liminf_{i\to\infty} c(i) = \infty\}$;
(iv) eventually above any number: $\mathrm{R}_{\mathrm{inf}=\infty} = \{c \mid \liminf_{i\to\infty} c(i) = \infty\}$;
(v) unbounded: $\mathrm{R}_{\mathrm{sup}=\infty} = \{c \mid \limsup_{i\to\infty} c(i) = \infty\}$;
(vi) complete: $\mathrm{R}_{\mathrm{onto}} = \{c \mid \mathrm{range}(c) = \mathbb{N}\}$.

By requiring a learner to succeed regardless of what counter was chosen from the counter type, we can provide certain beneficial properties of a counter, while not providing others. For example, counters from $\mathrm{R}_{\mathrm{onto}}$ provide a complete enumeration of all natural numbers, but do not allow one to infer the number of data items seen so far. We illustrate the inclusion properties of the different sets of counters with the following diagram (inclusions are top to bottom; thus, inclusions of learning power when such counters are used are bottom to top).

$$\begin{tikzcd}
& \mathrm{Id} \arrow[dl, dash] \arrow[ddr, dash] & \\
\vec{\mathrm{R}}^! \arrow[d, dash] & & \\
\vec{\mathrm{R}} \arrow[d, dash] & & \mathrm{R}_{\mathrm{onto}} \arrow[ddl, dash] \\
\mathrm{R}_{\mathrm{inf}=\infty} \arrow[dr, dash] & & \\
& \mathrm{R}_{\mathrm{sup}=\infty} \arrow[d, dash] & \\
& \perp &
\end{tikzcd}$$

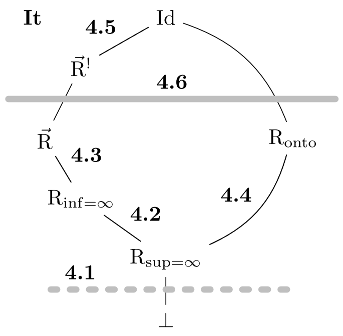
The symbol $\perp$ denotes no use of counter. The weakest type of counter is $\mathrm{R}_{\mathrm{sup}=\infty}$, the unbounded counter. The advantage over having no counter at all is to be able to make computations with higher and higher time bounds; in fact, it is easy to see that set-driven learning merely requires a counter from $\mathrm{R}_{\mathrm{sup}=\infty}$ to gain the full power of TxtEx-learning. John Case pointed out that any text for an infinite language implicitly provides a counter from $\mathrm{R}_{\mathrm{sup}=\infty}$.

A somewhat stronger counter type is $\mathrm{R}_{\mathrm{inf}=\infty}$; the intuitive advantage of this counter is that a learner will not repeat mistakes made on small counter values indefinitely, but only the behavior on large counter values affects the learning process in the limit. For the monotone counters from $\vec{\mathrm{R}}$, the advantage is again that early mistakes are not repeated once learning has proceeded to a later stage (as in, higher counter value), as well as a monotonicity in advancing through

[2] The counter types (i), (iii) and (v) were suggested by John Case in private communication.
[3] "Id" stands for identity; $\mathbb{N}$ denotes the natural numbers and $\mathrm{id}_{\mathbb{N}}$ the identity on $\mathbb{N}$.

these stages. Counters from $\vec{R}^!$ have the additional benefit of providing an upper bound on the number of examples seen so far. Id is the strongest type of counter providing exactly the number of data elements presented so far. Also, all natural numbers are listed, which allows a learner to divide up tasks into infinitely many subtasks to be executed at the corresponding counter value; the counter type R_{onto} models this latter advantage while dropping the order restriction.

The main results of this paper consider iterative learning and are as follows. Even adding the weakest type of counter, $R_{\text{sup}=\infty}$, to plain iterative learning gives an increase in learning power; however, there is no increase on learning classes of infinite languages only (see Theorem 4.1). Furthermore, the criteria corresponding to the six counter types are divided into two groups of criteria of equal learning power as depicted by the following diagram. The gray line divides the two groups, the dashed gray line indicates separation only on sets of languages containing finite sets; the numbers represent the numbers of the corresponding theorems.



In particular, only the strict monotonicity of a counter gives additional learning power over $R_{\text{sup}=\infty}$ counters. The proofs for the claims inherent in the diagram can be found in Section 4.

Theorem 4.6 in Section 4 shows the separation depicted in the above diagram; its proof uses a self-learning class of languages [7,8] and Case's *Operator Recursion Theorem* (ORT) [2,13].

Extending these results to settings where learners have additional resources is ongoing work; preliminary results show that when adding a finite number of memory states, we get a similar diagram as for iterative learning above.

One may wonder whether some two of the counter types introduced above always yield the same learning power (as many did in the case of iterative learning), across all possible settings. In Section 1.1 we discuss why this is not the case. The associated proofs can be found in Section 5.

Section 2 gives some mathematical preliminaries. In Section 3 we establish that any separation of learning criteria power will necessarily be witnessed by a class containing only infinite languages, if the considered learning criteria have access to any of the six counter types. As already mentioned, Section 4 gives some details for the diagram above, and Section 5 gives proofs for claims from Section 1.1.

The present paper is an extension of the conference paper [15].

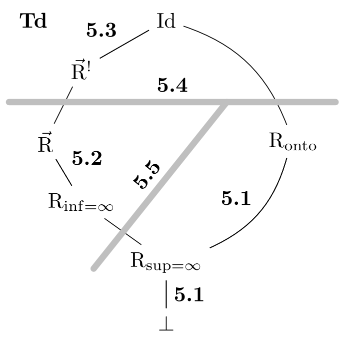
1.1. *Differences in counters*

In this section we show that, for any choice of two different counter types, there is a learning criterion which, when augmented with one of the counter types, yields different classes of languages learnable than when augmented with the other. We already saw some such separations in the setting for iterative learning. Now we will give some other settings witnessing other separations.

First, consider iterative learning with one additional feedback query (see [17,5]). In this setting, in each iteration, the learner may ask about one datum whether it has been presented previously. Frank Stephan and Sanjay Jain (private communication) have a proof that, in this setting, there are classes of languages learnable with R_{onto} counters which are not learnable with $\vec{R}^!$ counters. Thus, there are settings where Id separates from $\vec{R}^!$, and where R_{onto} separates from $R_{\text{sup}=\infty}$.

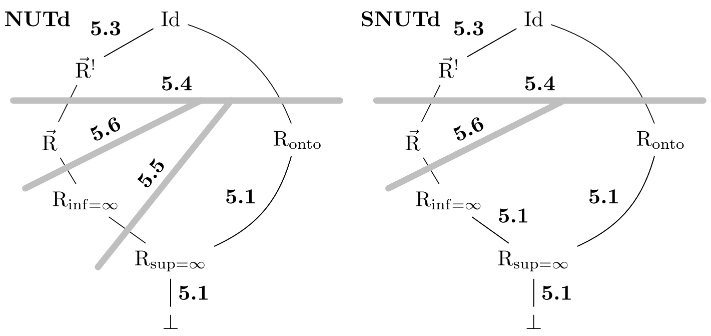
For more separations, we turn to very simple learning criteria. We consider *transductive* learning (**Td**), that is, learning without memory (which equals a degenerate case of memoryless learning with bounded memory states, where the bound on the number of states is 1; [4,6]). In this somewhat artificial toy setting a learner is presented with a datum (and possibly a counter value) in each iteration, and not more. Note that learners are allowed to output the special symbol ? to, in effect, keep the previous conjecture as the latest guess.

It is not hard to see that, for transductive learning, adding an $R_{\text{sup}=\infty}$ or R_{onto} counter does not improve learning power. However, other types of counter do provide increases. The general result is depicted in the following diagram, using the same format as in the diagram on iterative learning above.



The intuitive reasons for the separations are as follows. An infinite limit inferior guarantees that mistakes on early counter values are not repeated infinitely often. With a strictly monotone counter, any mistake on a counter value z is guaranteed to be preceded by at most z other data items; thus, if the language contains at least $z+1$ data items giving the correct output, the mistake will be rectified.

The situation changes if we require of the learner additionally to never abandon correct conjectures – either only not semantically (called non-U-shaped learning, **NU**, [1]) or not even syntactically (strongly non-U-shaped learning, **SNU**, [9]). The resulting groupings and separations are depicted in the following two diagrams.



Intuitively, for learning criteria requiring non-U-shapedness, order plays an important role (wrong conjectures may only come before correct ones), leading to the separations between $R_{inf=\infty}$ and $\vec{R}$. For strongly non-U-shaped learning with $R_{inf=\infty}$ counter, a learner may not give two different conjectures for any two pairs of datum/counter value.

All the above settings together show that, for each two different types of counter, there are settings of associated learning criteria where the learning power separates.

2. Mathematical preliminaries

Unintroduced notation follows [19].

$\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$. 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.

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 fix any computable 1–1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[4] 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., $\exists x_0\colon\ f(x_0) = p \wedge \forall x \geqslant x_0\colon\ f(x)\downarrow \in \{?, p\}$.[5]

$\mathcal{P}$ and $\mathcal{R}$ denote, respectively, the set of all partial computable and the set of all computable functions (mapping $\mathbb{N} \to \mathbb{N}$).

[4] For a linear-time example, see [18, Section 2.3].
[5] $f(x)$ converges should not be confused with f converges *to*.

We let φ be any fixed acceptable programming system for $\mathcal{P}$. 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.

In this paper, an *operator* is a mapping from any fixed number of arguments from $\mathcal{P}$ into $\mathcal{P}$.

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 $f(0), \dots, f(i-1)$ (undefined, if one of these values is undefined).

For one of the proofs we will use Case's *Operator Recursion Theorem* (**ORT**), providing *infinitary* self-and-other program reference [2,3,13]. **ORT** itself states that, for all operators Θ there are f with $\forall z$: $\Theta(\varphi_z) = \varphi_{f(z)}$ and $e \in \mathcal{R}$,

$$\forall a, b\colon\ \varphi_{e(a)}(b) = \Theta(e)(a, b). \tag{1}$$

2.1. Learning criteria

In this section we formally introduce our setting of learning in the limit and associated learning criteria. We follow [14] in its "building-blocks" approach for defining learning criteria. This approach aims at avoiding confusion by clearly stating all requirements of a learning criterion; furthermore, it enables us to define many learning criteria with a few definitions of "building-blocks" composed in different ways. As an additional benefit, we will be able to give theorems which apply to large classes of learning criteria (see Section 3).

A *learner* is a partial computable function from $\mathbb{N}$ to $\mathbb{N} \cup \{?\}$. 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. We will extend the notion of texts to include counters as follows.

A *counter* is a function $c : \mathbb{N} \to \mathbb{N}$; a *type of counter* is a set of counters R. For any type of counter R, we let $\mathbf{TxtCtr}[R]$ be the set of all functions $\langle T, c\rangle = \lambda i.\langle T(i), c(i)\rangle$ with T a text and $c \in R$. We call an element from $\mathbf{TxtCtr}[R]$ a *text/counter*, and the content of any text/counter is the content of its text component.

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

We define the sequence generating operators **It** and **Td** (corresponding to the learning criteria discussed in the introduction) as follows. For all learners h, text/counters T^c and all i,

$$\mathbf{It}\big(h, T^c\big)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0;^{6} \\ h(\mathbf{It}(h, T^c)(i-1), T^c(i-1)), & \text{otherwise;} \end{cases}$$

$$\mathbf{Td}\big(h, T^c\big)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0; \\ \mathbf{Td}(h, T^c)(i-1), & \text{else, if } h(T^c(i-1)) = ?; \\ h(T^c(i-1)), & \text{otherwise.} \end{cases}$$

Thus, in iterative learning, the learner has access to the previous conjecture, but not so in transductive learning. However, in transductive learning, the learner can *implicitly* take over the previous conjecture by outputting "?".

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/counter. We give the examples of explanatory (**Ex**), non-U-shaped (**NU**) and strongly non-U-shaped (**SNU**) learning, which were discussed in Section 1. Formally, we let, for all conjecture sequences p and text/counters T^c,

$$\mathbf{Ex}\big(p, T^c\big) \Leftrightarrow \big[\exists n_0, q\colon\ p \text{ converges to } q \wedge \mathrm{content}\big(T^c\big) = W_q\big];$$

$$\mathbf{NU}\big(p, T^c\big) \Leftrightarrow \big[\forall i\colon\ W_{p(i)} = \mathrm{content}\big(T^c\big) \Rightarrow (\forall j \geqslant i\colon\ W_{p(j)} = W_{p(i)})\big];$$

$$\mathbf{SNU}\big(p, T^c\big) \Leftrightarrow \big[\forall i\colon\ W_{p(i)} = \mathrm{content}\big(T^c\big) \Rightarrow \big(\forall j \geqslant i\colon\ p(j) = p(i)\big)\big].$$

We combine any two sequence acceptance criteria δ and δ' by conjuncting them; we denote this by juxtaposition (for example, **SNU** and **NU** are meant to be always used together with **Ex**). With **T** we denote the always true sequence acceptance criterion (no restriction on learning).

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

We call a sequence acceptance criteria δ *delay invariant* iff,

- for all monotone and unbounded f, and p, T^c, $\delta(p, T^c)$ implies $\delta(p \circ f, T^c)$;
- for all p, T^c and p' such that $p'(0) = ?$ and $\lambda n \centerdot p'(n+1) = p$, $\delta(p, T^c)$ implies $\delta(p', T^c)$.

Note that any combination of **Ex**, **NU** and **SNU** is delay invariant.[7]

For any set of text/counters α, any sequence generating operator β and any combination of sequence acceptance restrictions δ, $\alpha\beta\delta$ is a *learning criterion*. A learner h *$\alpha\beta\delta$-learns* the set

$$\alpha\beta\delta(h) = \{L \in \mathcal{E} \mid \forall T^c \in \alpha\text{: content}(T^c) = L \Rightarrow \delta(\beta(h, T^c), T^c)\}.$$

Abusing notation, we also use $\alpha\beta\delta$ to denote the set of all $\alpha\beta\delta$-learnable classes (learnable by some learner).

3. Separations by classes of infinite languages

In this section we show that, for iterative learning, all separations between the learning criteria corresponding to the different counter types are necessarily witnessed by sets of infinite languages. The reasoning for this can be extended to include many other learning criteria.

For an operator Θ, a learning criterion I is called Θ-robust iff, for any class of languages $\mathcal{L}$, I-learnability of $\mathcal{L}$ is equivalent to I-learnability of $\Theta(\mathcal{L})$ (elementwise application of Θ).[8] We let Θ_0 be the mapping $L \mapsto 2L \cup (2\mathbb{N}+1)$. Obviously, there is a function f_0 such that $\forall e$: $\Theta_0(W_e) = W_{f_0(e)}$. Note that Θ_0 has an inverse Θ_0^{-1} for which a function analogous to f_0 exists.

Theorem 3.1. *Let $R \in \{\mathrm{R}_{\sup=\infty}, \mathrm{R}_{\inf=\infty}, \vec{\mathrm{R}}, \vec{\mathrm{R}}^!, \mathrm{Id}, \mathrm{R}_{\mathrm{onto}}\}$. Then we have that the learning criterion* **TxtCtr**[R]**ItEx** *is Θ_0-robust.*

Proof. Let $\mathcal{L} \in$ **TxtCtr**[R]**ItEx**. Obviously, $\Theta_0(\mathcal{L})$ can be learned using the learner for $\mathcal{L}$ by ignoring odd data (considering them as #) and halving all even data, mapping all conjectures with f_0.

Conversely, let a learner h_0 for $\Theta_0(\mathcal{L})$ be given. Consider first the case of $R = \mathrm{Id}$. Define the following function h'.

$$\forall e, x, z : h'(e, x, z) = h_0(h_0(e, 2x, 2z), 2z+1, 2z+1).$$

Intuitively, on a text T, h' simulates h_0 on the text where $2T$ is interleaved with odd data. We use 1-1 s-m-n to get a function to turn conjectures for a language from $\Theta_0(\mathcal{L})$ into the corresponding language from $\mathcal{L}$ (we use 1-1 so that we can extract and use the conjectures of h_0 from the previous generation as input to h_0), resulting in a learner h.

Note that, for $R = \mathrm{R}_{\mathrm{onto}}$, the above construction of h works just as well. All other cases are similar as follows.

For $R \in \{\mathrm{R}_{\sup=\infty}, \mathrm{R}_{\inf=\infty}, \vec{\mathrm{R}}\}$, when we see counter value of z, we simulate h_0 on all odd data $\leqslant z$ and on the current datum times two, using a counter value of z for all of them.

For $R = \vec{\mathrm{R}}^!$, when we see counter value of z, we simulate h_0 on all odd data $< z$ and on the current datum times two, using a counter value of $z^2 + i$ for the ith run of h_0. Thus, within these batches of data, the counter values are strictly increasing. The next batch will start with a counter value of $(z+1)^2 = z^2 + 2z + 1$. This exceeds the last counter used in the previous batch, as the previous batch had a size $\leqslant z+1$. □

Theorem 3.2. *Let I and I' be Θ_0-robust learning criteria. Then I and I' separate in learning power iff they separate on classes of infinite languages.*

Proof. Suppose a class of languages $\mathcal{L}$ separates I and I'. Then $\Theta_0(\mathcal{L})$, a class of infinite languages, also witnesses this separation, as I and I' are Θ_0-robust. The converse is trivial. □

From what we saw in this section we get the following corollary.

Corollary 3.3. *Let $R, R' \in \{\mathrm{R}_{\sup=\infty}, \mathrm{R}_{\inf=\infty}, \vec{\mathrm{R}}, \vec{\mathrm{R}}^!, \mathrm{Id}, \mathrm{R}_{\mathrm{onto}}\}$. Then the learning criteria* **TxtCtr**[R]**ItEx** *and* **TxtCtr**[R']**ItEx** *separate iff the separation is witnessed by a class of infinite languages. Furthermore, it is witnessed by a class of languages all containing all odd numbers.*

[7] Intuitively, delay invariance means that a sequence of conjectures remains a valid sequence for a given text if it is modified by repeating or omitting conjectures, or by delaying the next conjecture with ?; however, the order of the conjectures may not be changed.

[8] Case and Kötzing [8] explore some notions of robustness for function learning.

4. Comparison of counter types

In this section we present the proofs for the results regarding iterative learning with counter. First we compare the weakest counter with no counter at all (see Theorem 4.1). Theorems 4.2 through 4.5 give equivalences of learning power as indicated in Section 1. Finally, Theorem 4.6 gives the separation between strictly monotone counters and weaker counters.

Looking into the proof of Theorem 4 in [10] (showing that an Id counter allows one to learn languages which cannot be learned set-drivenly), we see that even the counters from $\mathrm{R}_{\sup=\infty}$ allow one to learn more than learning set-drivenly (and, thus, than what can be learned iteratively without a counter; this was shown in [16], see also [10] for a discussion on the relation between set-driven, iterative and iterative with Id counter learning). This leads to the first part of the next theorem. However, this proof makes use of finite languages. John Case, in private communication, remarked that $\mathrm{R}_{\sup=\infty}$-counters are provided by texts for infinite languages for free, leading to the second part of the theorem. We let $\mathcal{E}_\infty$ denote the set of all *infinite* $\mathtt{ce}$ sets.

Theorem 4.1. *We have*

$$\mathbf{TxtItEx} \subset \mathbf{TxtCtr}[\mathrm{R}_{\sup=\infty}]\mathbf{ItEx}$$

and

$$\mathrm{Pow}(\mathcal{E}_\infty) \cap \mathbf{TxtItEx} = \mathrm{Pow}(\mathcal{E}_\infty) \cap \mathbf{TxtCtr}[\mathrm{R}_{\sup=\infty}]\mathbf{ItEx}.$$

For the next step up the hierarchy of counter types, we don't get an increase in learning power.

Theorem 4.2. *We have*

$$\mathbf{TxtCtr}[\mathrm{R}_{\sup=\infty}]\mathbf{ItEx} = \mathbf{TxtCtr}[\mathrm{R}_{\inf=\infty}]\mathbf{ItEx}.$$

Proof. Clearly we get "$\subseteq$". The intuitive reason for the inclusion "$\supseteq$" is as follows. We can use the max of counter value, hypothesis and datum as a new counter to work with. If the conjecture changes infinitely often, then the new counter is from $\mathrm{R}_{\inf=\infty}$. Hence, the learning will converge; furthermore, for any fixed number z, only finitely many data points are evaluated with a counter value below z.

Let $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{R}_{\inf=\infty}]\mathbf{ItEx}$ as witnessed by h_0. By Corollary 3.3, we can assume, without loss of generality, that $\mathcal{L}$ contains only infinite languages. Obviously, using standard padding arguments, we can assume the sequence of h_0's conjectures, on any text, to be non-decreasing in numeric value. Furthermore, we can assume that, whenever h_0 would make a mind change when the present datum was replaced with a #, then it would also change its mind on the actual datum.

Let h be such that

$$h(\emptyset) = h_0(\emptyset);$$
$$\forall e, x, z : h(e, x, z) = h_0\big(e, x, \max(e, x, z)\big).$$

That is, h has the same initial conjecture as h_0 and uses the maximum of current conjecture, current datum (we let # count as 0) and current counter value as new counter value.

Let $L \in \mathcal{L}$, T a text for L and $c \in \mathrm{R}_{\sup=\infty}$ a counter. Suppose, by way of contradiction, h on T and c does not converge. Then h simulates h_0 on T and a counter from $\mathrm{R}_{\inf=\infty}$; this converges, a contradiction.

Suppose, by way of contradiction, h on T and c does converge, but not to an index for L. We focus on the text after h's convergence to some (wrong) conjecture e.

Consider first the case where there is a finite s such that, for all $s' \geqslant s$, $h_0(e, \#, s') \neq e$, that is, h_0 changes its mind on # for all but finitely many counter values. Then, by one of our assumptions on h_0, at some point after the convergence of h on T, we get a counter value $\geqslant s$ so that h will change its mind, a contradiction.

Consider now the case where, for infinitely many s, $h_0(e, \#, s) = e$. Let T' be the text derived from T where we do not change anything before the point of h's convergence on T, and afterwards replace all repeated data with #es. Let c' be such that, for all i, $c'(i) = \max_t[h_0(e, T'(i), t) = e]$ (possibly ∞) – that is, c' denotes the maximum counter value for h_0 to not change its mind. As e is incorrect and needs to be changed by h_0 eventually on any counter with infinite limit inferior, c' has *finite* limit inferior. Thus, there is a bound s such that, for infinitely many i, $\max_t[h_0(e, T'(i), t) = e] \leqslant s$. Because of the case we consider now, we know that there are infinitely many i with $T'(i) \neq \#$ and $\max_t[h_0(e, T'(i), t) = e] \leqslant s$. One of these pairwise different $T'(i) = T(i)$ will be larger than s, leading to a mind change with h, a contradiction. □

Note that the just above proof is not entirely a simulation argument: h_0 is being simulated, but not on counters for which we have immediate performance guarantees.

Also the next step in the counter hierarchy does not yield a difference in learning power.

Theorem 4.3. *We have*

$$\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{ItEx} = \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{ItEx}.$$

Proof. Clearly we get "$\subseteq$". Let $\mathcal{L} \in \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{ItEx}$ as witnessed by h_0. Obviously, using standard padding arguments, we can assume the sequence of h_0's conjectures, on any text, to be non-decreasing in numeric value. Furthermore, we can assume each conjecture to exceed the counter value on which it was first output.

For all e, x, z, we let $f(e,x,z)$ be the least t with $e \leqslant t \leqslant \max(e,z)$ and $h_0(e,x,t) \neq e$, if existent (undefined otherwise). Note that the domain of f is decidable.

Let h be such that, for all e, x, z,

$$h(\emptyset) = h_0(\emptyset);$$
$$h(e,x,z) = \begin{cases} h_0(e,x,f(e,x,z)), & \text{if } f(e,x,z)\downarrow, \\ e, & \text{otherwise.} \end{cases}$$

Let $L \in \mathcal{L}$, T a text for L and $c \in \mathrm{R}_{\mathrm{inf}=\infty}$ a counter. We define a counter c' on argument i thus. Let e be the conjecture of h after $T[i]$; if, in the definition of $h(e,T(i),c(i))$, the first case holds, then $c'(i) = f(e,T(i),c(i))$. Otherwise, if there will be a mind change of h on T with counter c later (i.e., on a counter value of at least e), then $c'(i) = e$, else $c'(i) = \max(e, \min\{c(j) \mid j \geqslant i\})$.

It is easy to see that $c' \in \vec{\mathrm{R}}$ and h on T and c simulates h_0 on T and c'. $\square$

Note that, in the proof just above, the argument is again not entirely a simulation: defining the counter c' requires knowledge of future mind changes and of infinitely many future counter values.

Next we show that complete counters do not give an advantage over $\mathrm{R}_{\mathrm{sup}=\infty}$ counters.

Theorem 4.4. *We have*

$$\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{sup}=\infty}]\mathbf{ItEx} = \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{onto}}]\mathbf{ItEx}.$$

Proof. The inclusion "$\subseteq$" is trivial. Suppose, by way of contradiction, a set $\mathcal{L}$ separates the two criteria considered by this theorem. Then, using Corollary 3.3, we get that a class of languages all containing all odd data witness the separation as well. From a text for such a language we can extract a complete counter (by dividing each datum by 2, rounding down), a contradiction. $\square$

Last we show that also the topmost step in the counter hierarchy gives no difference in learning power.

Theorem 4.5. *We have*

$$\mathbf{TxtCtr}[\vec{\mathrm{R}}^{!}]\mathbf{ItEx} = \mathbf{TxtCtr}[\mathrm{Id}]\mathbf{ItEx}.$$

Proof. Clearly we get "$\subseteq$". The intuitive idea for "$\supseteq$" is as follows. The learner can store in the conjecture the last counter on which it changed the conjecture and fill up all the gaps in between two counter values with #es.

Let $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{Id}]\mathbf{ItEx}$ as witnessed by h_0. Without loss of generality, we assume that h_0 will change its mind on any datum whenever it would change its mind on a # (this is not as trivial as for other counter types, but straightforward to show). Using 1-1 s-m-n, we fix any 1-1 function pad such that, for all e, x, $W_{\mathrm{pad}(e,x)} = W_e$. We use this function for a learner to memorize certain information (at the cost of a mind change).

We define a function h_0^* inductively as follows. For all e, z,

$$h_0^*(e,\emptyset,z) = e;$$
$$\forall \sigma, x\colon\ h_0^*(e,\sigma x,z) = h_0\bigl(h_0^*(e,\sigma,z), x, z + \mathrm{len}(\sigma)\bigr).$$

Let h be such that, for all e, x, z, z' with $z > z'$,

$$h(\emptyset) = \mathrm{pad}\bigl(h_0(\emptyset), 0\bigr);$$
$$h\bigl(\mathrm{pad}\bigl(e,z'\bigr),x,z\bigr) = \begin{cases} \mathrm{pad}(h_0^*(e, \#^{z-z'-1}x, z'), z), & \text{if } h_0^*(e, \#^{z-z'-1}x, z') \neq e; \\ \mathrm{pad}(e,z'), & \text{otherwise.} \end{cases}$$

Let $L \in \mathcal{L}$, T a text for L and $c \in \vec{\mathrm{R}}^{!}$ a counter. We define a text T' thus.

$$\forall i\colon\ T'(i) = \begin{cases} T(k), & \text{if } i = c(k); \\ \#, & \text{otherwise.} \end{cases}$$

Clearly, T' is a text for L and $T' \circ c = T$. Let p be the sequence of outputs of h on T and p' the sequence of outputs of h_0 on T'. Now we have $p' \circ c = p$, as h makes mind changes on data whenever it would make a mind change on a pause with the same counter value. $\square$

The next theorem shows that the remaining two classes of learning power do separate. The proof is highly technical and makes use of a self-learning class of languages – the two learning criteria are very similar and apparently hard to separate.

Theorem 4.6. *We have*

$$\mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{ItEx} \subset \mathbf{TxtCtr}[\vec{\mathrm{R}}^!]\mathbf{ItEx}.$$

Proof. The inclusion is trivial. The intuitive idea of the separation is as follows. We use a self-learning class of languages (see the definition of $\mathcal{L}$ below for an example of a self-learning class; these classes are discussed in more detail in [7,8]). The main idea is to code into the data what the learner should say; if given appropriate counter, this will be successful.

For our proof, we will try to generate data $b(0), \dots, b(c_0 - 1)$ and data $a(0), \dots$ and having the learner learn both the set of all a-data and a mix of finitely many a- and b-data. Also, we require that, on b-data with a small counter, the learner makes no mind change. With a strictly increasing counter, some b-data will be presented with a large enough counter, while the same is not true for arbitrary monotone counters. The following diagram depicts the two different cases for different counter types. In the first row different counter values are given, the second and third rows show texts for the language of mixed a- and b-data. Strict counters are allowed only one datum per counter value, monotone are allowed any (finite) number of data per counter value (this is meant in the column for counter value 0 with $b[c_0]$: all data $b(i)$ for $i < c_0$ are presented in order with counter value 0).

counter	0	1	2		$c_0 - 1$	c_0	
$\vec{\mathrm{R}}^!$:	$b(0)$	$b(1)$	$b(2)$	$\dots$	$b(c_0 - 1)$	$a(0)$	$\dots$
$\vec{\mathrm{R}}$:	$b[c_0]$	$a(0)$	$a(1)$	$\dots$	$a(c_0 - 2)$	$\dots$	

The proof of Claim 7 will make use basically of the second text/counter to derive a contradiction, along with a text/counter for only a-data where the counter value 0 has no data; these two text/counters will be indistinguishable for h.

However, since learner h may change its mind on early b-data, we will start the text with as much b-data (and higher and higher counter values) until we do not see a mind change any more on b-data (we will make sure that this has to happen).

We now turn to formalizing the many ingredients for the proof, including the computability of the many steps outlined above. We start the formal argument with many definitions; the reader is advised to only skim these definitions and verify the claims of computability. The argument is easier to understand when reading the sequence of claims following the definitions.

Let $h_0 \in \mathcal{P}$ be such that h_0's initial conjecture is ? and, for all e, x, c,

$$h_0(e, x, c) = \begin{cases} e, & \text{if } x = \#; \\ \varphi_x(e, c), & \text{otherwise.} \end{cases}$$

Let $\mathcal{L} = \mathbf{TxtCtr}[\vec{\mathrm{R}}^!]\mathbf{ItEx}(h_0)$. Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{ItEx}$, as witnessed by $h \in \mathcal{P}$. Without loss of generality, we can assume that h changes its mind on any datum/counter if it would change its mind on a pause with the same counter value.

We define h^* as a function on sequences inductively as follows.

$$h^*(\emptyset) = h(\emptyset);$$

$$\forall \sigma, x\colon\ h^*(\sigma x) = h\big(h^*(\sigma), x, \mathrm{len}(\sigma)\big).$$

We make a number of definitions depending on some $a, b \in \mathcal{R}$ and $d \in \mathbb{N}$. We will not display this dependence in notation and later define what a, b and d are. We abbreviate, for all k, $a_k = \lambda i \centerdot a(k, i)$ and $b_k = \lambda i \centerdot b(k, i)$. We fix a program for h^* and let, for all σ and c, $h^*(\sigma)\downarrow_c$ denote that this program on σ converges in c steps. We define the following computable predicate P.

$$\forall \sigma, k, c\colon\ P(\sigma, k, c) \quad \Leftrightarrow \quad \exists i \leqslant c\colon\ h^*\big(\sigma a_k[i]\big)\downarrow_c = h^*\big(\sigma a_k[i+1]\big)\downarrow_c.$$

Let σ_0 be the sequence only listing d; for all k, let $\sigma_{k+1} = \sigma_k b_k(c)$ with c minimal such that h^* makes a mind change on $\sigma_k b_k(c)$ and $\neg P(\sigma_k, k, c)$ (undefined, if no such c exists, or if the computation of h^* goes undefined while trying to find one). We let $B = \bigcup_{k \in \mathbb{N}} \mathrm{content}(\sigma_k)$.

Furthermore, we abbreviate, for all k, i and c,

$$\begin{aligned} Q_k(i) &\Leftrightarrow h^*(\sigma_k a_k[i])\downarrow = h^*(\sigma_k a_k[i+1])\downarrow; \\ \overline{Q}_k(i) &\Leftrightarrow h^*(\sigma_k a_k[i])\downarrow \neq h^*(\sigma_k a_k[i+1])\downarrow; \\ P_k(c) &\Leftrightarrow P(\sigma_k, k, c). \end{aligned}$$

Note that Q and $\overline{Q}$ are partial recursive. If there are k and c such that

$$P_k(c) \wedge \forall i < c\colon h^*(\sigma_k)\downarrow = h^*(\sigma_k b_k(i))\downarrow, \tag{2}$$

then we let k_0 and c_0 be the example which minimizes c (note that there can be only one k such that Eq. (2) holds: if Eq. (2) holds for some k and c, then σ_{k+1} is not defined because of the condition $\neg P_k(c)$). We let $A_k = \{a_k(i) \mid \forall j \leqslant i\colon \overline{Q}_k(j)\}$, and $A_k^+ = \{a_k(i) \mid \forall j < i\colon \overline{Q}_k(j)\}$. If c_0 and k_0 are defined, we let $B^+ = B \cup \{b_{k_0}(i) \mid i < c_0\}$; otherwise $B^+ = \emptyset$. Note that, given a, b, d and k, A_k, A_k^+, B and B^+ are ce.

We are now ready to specify a, b and d. By the Operator Recursion Theorem (see Eq. (1)), there are 1-1 $a, b, q \in \mathcal{R}$ and $p, r, d \in \mathbb{N}$ such that, for all c, e, i, k

$$\begin{aligned} &W_p = \{x \in A_k \mid k_0 \text{ is defined and equal } k\} \cup B; \\ &W_{q(k)} = A_k \cup B; \\ &W_r = \left\{x \in A_k^+ \;\middle|\; k_0 \text{ is defined and equal } k\right\} \cup B^+; \\ &\varphi_{a_k(i)}(e,c) = \begin{cases} \uparrow, & \text{if } \sigma_k \text{ is not defined;} \\ q(k), & \text{if } e \in \{p, ?\} \text{ and } \neg P_k(c); \\ r, & \text{if } e \in \{p, ?\}, P_k(c) \text{ and } i = \mu j . Q(j);^{9} \\ e, & \text{otherwise;} \end{cases} \\ &\varphi_{b_k(i)}(e,c) = \begin{cases} r, & \text{if } e = q(k); \\ e, & \text{otherwise;} \end{cases} \\ &\varphi_d(e,c) = \begin{cases} p, & \text{if } e = ?; \\ e, & \text{otherwise.} \end{cases} \end{aligned}$$

Claim 1. *Let k be such that σ_k is defined. Then, for all sequences ρ with elements from* $\mathrm{range}(a_k) \cup \mathrm{range}(b)$, *$h^*(\rho)\downarrow$.*

Proof. Let d' and s be such that, for all e, c, $\varphi_{d'}(e,c) = s$ and $W_s = \mathrm{range}(a_k) \cup \mathrm{range}(b) \cup \{d'\}$. Then we have that h_0 on any text for W_s converges to s and, thus, learns W_s. Now we have $W_s \in \mathcal{L}$ and therefore, for all sequences ρ with elements from W_s, $h^*(\rho)\downarrow$. □

Claim 2. *B is finite.*

Proof. Suppose, by way of contradiction, B is infinite. Then, for all k, σ_k is defined and h^* and the text $\bigcup_k \sigma_k$ for B makes infinitely many mind changes. Furthermore, from Eq. (2) we see that k_0 and c_0 are not defined and, thus, $A_{k_0} = \emptyset$. As h_0 on any text for B converges to p (thanks to $d \in B$), and, thus, correctly identifies B, we have $B \in \mathcal{L}$. However, as stated before, h on text $\bigcup_k \sigma_k$ for B makes infinitely many mind changes and fails to learn B, a contradiction. □

From the previous claim there is k_1 such that σ_{k_1} is defined and σ_{k_1+1} is not. Furthermore,

$$B \subseteq \{d\} \cup \bigcup_{j<k_1} \mathrm{range}(b_j). \tag{3}$$

This shows that elements from B do not make h_0 make a mind change when $q(k_1)$ is the latest conjecture.

Claim 3. *There is a c such that $P_{k_1}(c)$.*

Proof. Suppose, by way of contradiction, for all c, $\neg P_{k_1}(c)$. Thus, A_{k_1} is infinite, and $W_{q(k_1)} \in \mathcal{L}$, but h, on the text $\sigma_{k_1} a_{k_1}$ with Id-counter does not converge to an index for $W_{q(k_1)}$, a contradiction. □

[9] Note that $(P(c) \wedge i = \mu j . Q(j))$ is decidable, as, in case of $P(c)$, $\mu j . Q(j)$ is the first witness of $P(c)$.

Claim 4. *k_0 and c_0 are defined.*

Proof. Let c_1 be minimal such that $\neg P_{k_1}(c_1)$ (we know about the existence from Claim 3). We have that k_1 and c_1 fulfill Eq. (2) as otherwise σ_{k_1+1} would be defined. $\square$

We now get $k_1 = k_0$. From Claim 1 we have that, for all i, either $Q_{k_0}(i)$ or $\overline{Q}_{k_0}(i)$. Let $i_0 = \mu j . Q_{k_0}(j)$. Clearly, $A_{k_0} = \{a_{k_0}(i) \mid i < i_0\}$ and $A^+_{k_0} = \{a_{k_0}(i) \mid i \leqslant i_0\}$.

Claim 5. *We have $W_p = A_{k_0} \cup B \in \mathcal{L}$.*

Proof. Data from $B \setminus \{d\}$ do not do anything (as they are all from b_j for $j < k_0 = k_1$, see Eq. (3)); d will merely change from ? to p; and data from A_{k_0} will evaluate to q_{k_0}, an index for W_p, or repeat the previous conjecture, as $a_{k_0}(i_0) \notin W_p$. We use d to ensure that $W_p \neq \emptyset$, so that h_0 will have at least one datum to use. $\square$

The proof of the next claim exploits the fact that h has to learn from arbitrary monotone counters, but h_0 only has to be successful on strictly monotone counters.

Claim 6. *We have $W_r = A^+_{k_0} \cup B^+ \in \mathcal{L}$.*

Proof. Let a text T for W_r be given, as well as a strictly monotone increasing counter r. We show that h_0 on $\langle T, c\rangle$ converges to an index for W_r.

Consider first the case where all elements of range$(b_{k_0}) \cap B^+$, of which there are c_0, are presented before any element from $A^+_{k_0}$. Then, at some later point with counter value at least c_0 (as there were already c_0 elements presented),[10] $a_{k_0}(i_0)$ will be presented (recall that $i_0 = \mu j . Q(j)$), and h_0 will have converged to r.

Consider second the case where some element $a_{k_0}(j) \in A^+_{k_0}$ is presented before some element of range$(b_{k_0}) \cap B^+$. Then, if $j = i_0$, h has converged to r; otherwise the output of h will be $q(k)$, and the later element from range$(b_{k_0}) \cap B^+$ will then make h_0 output r. $\square$

We are now ready to show that h fails to learn $\mathcal{L}$.

Claim 7. *We have that h does* not *learn $W_r = A^+_{k_0} \cup B^+ \in \mathcal{L}$ or $W_p = A_{k_0} \cup B \in \mathcal{L}$ with counters from $\vec{\mathrm{R}}$.*

Proof. Consider the text/counter T as follows. Start with σ_{k_0} and an Id-counter. Then append $b_{k_0}[c_0]$ with counter constantly k_0. Continue with an Id-counter (ignoring the elements from range(b_{k_0})) and append $a[i_0+1]\#^\infty$. This is a text for W_r with a counter from $\vec{\mathrm{R}}$.

Consider also the text/counter $T' = \sigma_{k_0} a[i_0]\#^\infty$ and an Id-counter. This is a text for $W_{q(k_0)}$ with a counter from $\vec{\mathrm{R}}$.

Bringing everything together, in particular Eq. (2) and the definition of i_0, we get that h converges to the same number (if any) on T and T'. As T and T' are for different languages in $\mathcal{L}$, h cannot learn both. $\square$ $\square$

5. Transduction and counters

In this section we formally prove the results regarding transductive learning with counter; the results are summarized in the diagrams in Section 1.1.

Theorem 5.1. *We have*

$$\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{onto}}]\mathbf{TdEx} = \mathbf{TxtTdEx} = \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{SNUTdEx}.$$

Proof. We start with the first equality, where the inclusion "$\supseteq$" is trivial. Let $\mathcal{L}$ be $\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{onto}}]\mathbf{TdEx}$-learnable, as witnessed by h. Without loss of generality, we can assume h to output ? on # with any counter value. Otherwise, the number output on pause may be output infinitely often on learning any language, in which case $\mathcal{L}$ can have at most one element, which is clearly $\mathbf{TxtTdEx}$-learnable.

[10] This is exactly where we use the counter being strictly monotone increasing.

We claim that, for all $L \in \mathcal{L}$, there is a p such that

- $L = W_{h(\emptyset)}$ or $\exists x \in L \forall z\colon\ h(x,z) = p$;
- $\forall x \in L, z \in \mathbb{N}\colon\ h(x,z) \in \{?, p\}$.

Let L be given. Suppose, by way of contradiction, that we can get two different outputs p and p' on two datum/counter pairs, where the data is from L. Then there is a text/counter (with complete counter) for L, where h outputs both p and p' infinitely often, a contradiction. Thus, there is at most one p ever output on a datum from L. Suppose that, for each $x \in L$, there is a z such that $h(x,z) = ?$. Then we can list all $x \in L$ with counters such that h always outputs ? and fill up all counter values without associated $x \in L$ with #. As L is learned by h, $L = W_{h(\emptyset)}$. This shows the claim formalized in the list above.

Clearly, any h as in the claim might as well learn without the additional help of a counter.

Regarding $\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{SNUTdEx}$, it is easy to see that it includes all **TxtTdEx**-learnable classes (using the characterization of learnability as given by the above list).

Regarding the converse, suppose $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{SNUTdEx}$ as witnessed by h'. Without loss of generality suppose $\mathcal{L}$ has at least two elements. Now we have that h' on # with any counter outputs ?, as otherwise h' can return to this hypothesis on any text after a correct conjecture; this is a contradiction since h' is strongly non-U-shaped and $\mathcal{L}$ has at least two elements. Furthermore, note that any two syntactically different outputs of h' on two elements of a language from $\mathcal{L}$ lead to a syntactic U-shape on some text/counter, with the counter from $\mathrm{R}_{\mathrm{inf}=\infty}$.

Let h be a **TxtTdEx**-learner such that, for all x, $h(x) = h'(x,x)$ (using the datum as counter for simulating h'). Let $L \in \mathcal{L}$. Let T be the text/counter such that, for all x, $T(x) = x/x$ if $x \in L$ and $T(x) = \#/x$ otherwise. Then h' is successful on T; since h' does not make two syntactically different outputs, h learns L. □

Theorem 5.2. *We have*

$$\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{TdEx} = \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{TdEx}.$$

Proof. The inclusion "$\subseteq$" is trivial. Let $\mathcal{L}$ be $\mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{TdEx}$-learnable, as witnessed by h.

We show that h witnesses $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{TdEx}$. Let $L \in \mathcal{L}$, T a text for L and $c \in \mathrm{R}_{\mathrm{inf}=\infty}$. Permute $\langle T, c\rangle$ into a text/counter $\langle T', c'\rangle$ such that c' is non-decreasing. Note that h on $\langle T', c'\rangle$ converges to an index for L.

We distinguish two cases. Either h on $\langle T', c'\rangle$ makes infinitely many non-? outputs. Then h on $\langle T, c\rangle$, from some counter value on, makes outputs only one non-? output q; furthermore, q is correct and repeated infinitely often. Thus, h on $\langle T, c\rangle$ converges to q as well.

Otherwise h on $\langle T', c'\rangle$ makes only finitely many non-? outputs. Then all those finitely many outputs are correct, as we could permute all later elements before any given output (and decrease the counter value as required to retain monotonicity of the counter). Thus, h on $\langle T, c\rangle$ converges to an index for L. □

Theorem 5.3. *Let δ be a delay invariant sequence acceptance criterion.*[11] *We have*

$$\mathbf{TxtCtr}[\vec{\mathrm{R}}^!]\mathbf{Td}\delta\mathbf{Ex} = \mathbf{TxtCtr}[\mathrm{Id}]\mathbf{Td}\delta\mathbf{Ex}.$$

Proof. Let $\mathcal{L}$ be $\mathbf{TxtCtr}[\mathrm{Id}]\mathbf{Td}\delta\mathbf{Ex}$-learnable, as witnessed by h.

We first show that, without loss of generality, h never outputs non-? on # with any counter value. Obviously, h does not output a number on infinitely many counter values: otherwise, these numbers output on pause may be infinitely output on learning any language, in which case $\mathcal{L}$ can have at most one element.

Suppose now h outputs some non-? conjecture on a positive finite number of counter values, the maximum being at z_0. We define h' as follows.

$$h'(\emptyset) = h(\#, z_0);$$
$$\forall x, z : h'(x,z) = \begin{cases} ?, & \text{if } z \leqslant z_0 \text{ or } x = \#;\\ h(x, z+1), & \text{otherwise.}\end{cases}$$

Clearly, on any text T, h' on T (with Id-counter) has the same convergence behavior as h on the text T' derived from T by inserting a # at position z_0.

This shows that, without loss of generality, h outputs ? on # and any counter value.

Now it is clear that such an h learns just as well from $\vec{\mathrm{R}}^!$ counters, as the gaps can be imagined to be filled with #es. □

Theorem 5.4. *We have*

$$\mathbf{TxtCtr}[\vec{\mathrm{R}}^!]\mathbf{TdSNUEx} \setminus \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{TdEx} \neq \emptyset.$$

[11] See Section 2.1 for the definition of delay invariant sequence acceptance criteria.

Proof. Let h_0 be such that $h_0(\emptyset)$ is an index for $\emptyset$ and, for all x, z,

$$h_0(x,z) = \begin{cases} ?, & \text{if } x = \#; \\ \varphi_x(z), & \text{otherwise.} \end{cases}$$

Let $\mathcal{L} = \mathbf{TxtCtr}[\vec{\mathrm{R}}^!]\mathbf{TdSNUEx}(h_0)$. Now we suppose, by way of contradiction, that $\mathcal{L} \in \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{TdEx}$ as witnessed by h. Without loss of generality, on # and any counter value, h outputs ? (this follows like in the proof of Theorem 5.3). As $\emptyset \in \mathcal{L}$, we have that $h(\emptyset)$ is an index for $\emptyset$.

We fix a program for h and denote, for all x, y, z, with $h(x,y){\downarrow}_z$ that this program on input x, y terminates in z steps.

By ORT, there are $a, p, q_0, q_1 \in \mathbb{N}$ and 1-1 range-disjoint $b_0, b_1 \in \mathcal{R}$ such that, for all $j \in \{0,1\}$ and all i, z,

$$P(z) \Leftrightarrow \neg\big(h(a,0){\downarrow}_z \neq ?\big);$$
$$W_p = \{a\};$$
$$W_{q_j} = \{a\} \cup \big\{b_j(i) \mid \forall k < i\colon\ P(k)\big\};$$
$$\varphi_a(z) = \begin{cases} p, & \text{if } P(z); \\ ?, & \text{otherwise.} \end{cases}$$
$$\varphi_{b_j(i)}(z) = \begin{cases} q_j, & \text{if } \neg P(z); \\ ?, & \text{otherwise.} \end{cases}$$

Clearly, if not $h(a,0){\downarrow} \neq ?$, then $\{a\} \in \mathcal{L}$, but h is not successful on the text $a\#^\infty$. Thus, $h(a,0){\downarrow} \neq ?$. Clearly, we have now $W_{q_0}, W_{q_1} \in \mathcal{L}$, but h has the same limit behavior on both the text/counters such that first all members of W_{q_j} of the form $b_j(z)$ with counter 0 are listed, then $a/0$ and then infinitely many # with strictly increasing counter. □

Theorem 5.5. *We have*

$$\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{TdNUEx} \setminus \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{sup}=\infty}]\mathbf{TdEx} \neq \emptyset.$$

Proof. Let h_0 be such that $h_0(\emptyset) = ?$ and, for all x, z,

$$h_0(x,z) = \begin{cases} ?, & \text{if } x = \#; \\ \varphi_x(z), & \text{otherwise.} \end{cases}$$

Let $\mathcal{L} = \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{TdNUEx}(h_0)$. Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{sup}=\infty}]\mathbf{TdEx}$ as witnessed by h.

As in Theorem 5.4, we fix a program for h and denote, for all x, y, z, with $h(x,y){\downarrow}_z$ that this program on input x, y terminates in z steps.

By ORT, there are $a_0, a_1, b, p_0, p_1, q \in \mathbb{N}$ and a computable predicate P such that, for all $j \in \{0,1\}$ and all z,

$$P(z) \Leftrightarrow h(a_0,0){\downarrow}_z \neq h(a_1,0){\downarrow}_z;$$
$$W_{p_j} = \{a_j\} \cup \begin{cases} W_q, & \text{if } \exists z\colon\ P(z); \\ \emptyset, & \text{otherwise;} \end{cases}$$
$$W_q = \{a_0, a_1, b\};$$
$$\varphi_{a_j}(z) = \begin{cases} p_j, & \text{if not } P(z); \\ ?, & \text{otherwise;} \end{cases}$$
$$\varphi_b(z) = q.$$

Clearly, if not $h(a_0,0){\downarrow} \neq h(a_1,0){\downarrow}$, then $\{a_0\}, \{a_1\} \in \mathcal{L}$, but h is not successful on both texts $a_0\#^\infty$ and $a_1\#^\infty$. Thus, $h(a_0,0){\downarrow} \neq h(a_1,0){\downarrow}$. Clearly, we have now $W_q \in \mathcal{L}$, but h does not converge on any text/counters such that both $a_0/0$ and $a_1/0$ are listed infinitely often, a contradiction. □

Theorem 5.6. *We have*

$$\mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{TdSNUEx} \setminus \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{TdNUEx} \neq \emptyset.$$

Proof. Let h_0 be such that $h_0(\emptyset) = ?$ and, for all x, z,

$$h_0(x,z) = \begin{cases} ?, & \text{if } x = \#; \\ \varphi_x(z), & \text{otherwise.} \end{cases}$$

Let $\mathcal{L} = \mathbf{TxtCtr}[\vec{\mathrm{R}}]\mathbf{TdSNUEx}(h_0)$. Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{TdNUEx}$ as witnessed by h.

As in Theorem 5.4, we fix a program for h and denote, for all x, y, z, with $h(x,y){\downarrow}_z$ that this program on input x, y terminates in z steps.

By ORT, there are $a_0, a_1, a_2, p_0, p_1, q_0, q_1 \in \mathbb{N}$ and a 1-1 $b_0, b_1 \in \mathcal{R}$ and a computable predicate P such that, for all $j \in \{0,1\}$, and all i, z,

$$P_j(z) \Leftrightarrow h(a_j, 0){\downarrow}_z \neq ?;$$

$$W_{p_j} = \{a_j\};$$

$$W_{q_j} = \{a_0, a_1, a_2\} \cup \mathrm{range}(b_j);$$

$$\varphi_{a_j}(z) = \begin{cases} p_j, & \text{if } \neg P_0(z) \wedge \neg P_1(z); \\ ?, & \text{otherwise;} \end{cases}$$

$$\varphi_{b_j(i)}(z) = \begin{cases} q_j, & \text{if } P_0(z) \vee P_1(z); \\ ?, & \text{otherwise.} \end{cases}$$

We derive a contradiction in either of two cases. First, suppose $\forall z$: $\neg P_0(z) \wedge \neg P_1(z)$. Then $\{a_0\}, \{a_1\} \in \mathcal{L}$, but h cannot be successful on both texts $a_0\#^\infty$ and $a_1\#^\infty$, a contradiction.

Now suppose $\exists z$: $P_0(z) \vee P_1(z)$. Without loss of generality, suppose $\exists z$: $P_0(z)$, i.e., $h(a_0,0){\downarrow} \neq ?$. We have now $W_{q_0}, W_{q_1} \in \mathcal{L}$, but h returns to the conjecture $h(a_0,0){\downarrow} \neq ?$ on texts for both W_{q_0} and W_{q_1} after the output of a correct conjecture when seeing $a_0/0$; thus, non-U-shapedness is violated in at least one of the cases. $\square$

6. Conclusion

In this paper we have seen an analysis of *why* an id iteration counter, as used in [10], increases the learning power of iterative learning: first, because the counter grows *unboundedly*, intuitively providing a growing time bound for computations; second, because the counter grows *strictly monotonically*, intuitively allowing for knowing an upper bound on how many data items have been presented previously.

We also saw the complete picture of the effect of counters in several settings concerning transductive learning, illustrating the usefulness of all different types of counter.

Furthermore, we saw in Section 3 a simple example for how the notion of robustness can help derive theorems with which many criteria (or pairs of criteria, like in Theorem 3.2) are covered at once. The author believes that future research would benefit from using similar approaches to derive general theorems.

Acknowledgements

The author was supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant No. NE 1182/5-1. Furthermore, the author would like to thank John Case, Sanjay Jain, Frank Stephan and Sandra Zilles for valuable and fruitful discussions, as well as the reviewers of both the conference version and this journal version of the paper, who provided excellent feedback. Finally, the author would like to thank Samuel E. Moelius III for inspiring discussions on the use of counters in limit learning. All this helped to greatly improve the paper.

References

[1] G. Baliga, J. Case, W. Merkle, F. Stephan, W. Wiehagen, When unlearning helps, Inf. Comput. 206 (2008) 694–709.
[2] J. Case, Periodicity in generations of automata, Math. Syst. Theory 8 (1974) 15–32.
[3] J. Case, Infinitary self-reference in learning theory, J. Exp. Theor. Artif. Intell. 6 (1994) 3–16.
[4] L. Carlucci, J. Case, S. Jain, F. Stephan, Results on memory-limited U-shaped learning, Inf. Comput. 205 (2007) 1551–1573.
[5] J. Case, S. Jain, S. Lange, T. Zeugmann, Incremental concept learning for bounded data mining, Inf. Comput. 152 (1999) 74–110.
[6] J. Case, T. Kötzing, Dynamic modeling in inductive inference, in: Proc. of ALT (Algorithmic Learning Theory), 2008, pp. 404–418.
[7] J. Case, T. Kötzing, Strongly non-U-shaped learning results by general techniques, in: Proc. of COLT (Conference on Learning Theory), 2010, pp. 181–193.
[8] J. Case, T. Kötzing, Measuring learning complexity with criteria epitomizers, in: Proc. of STACS (Symposium on Theoretical Aspects of Computer Science), 2011, pp. 320–331.
[9] J. Case, S. Moelius, Optimal language learning, in: Proc. of ALT (Algorithmic Learning Theory), 2008, pp. 419–433.
[10] J. Case, S. Moelius, U-shaped, iterative, and iterative-with-counter learning, Mach. Learn. 72 (2008) 63–88.
[11] R. Freivalds, E. Kinber, C. Smith, On the impact of forgetting on learning machines, J. ACM 42 (1995) 1146–1168.
[12] E. Gold, Language identification in the limit, Inf. Control 10 (1967) 447–474.
[13] S. Jain, D. Osherson, J. Royer, A. Sharma, Systems that Learn: An Introduction to Learning Theory, second edition, MIT Press, Cambridge, MA, 1999.
[14] T. 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.
[15] T. Kötzing, Iterative learning from positive data and counters, in: Proc. of ALT (Algorithmic Learning Theory), 2011, pp. 40–54.
[16] E. Kinber, F. Stephan, Language learning from texts: Mind changes, limited memory and monotonicity, Inf. Comput. 123 (1995) 224–241.
[17] S. Lange, T. Zeugmann, Incremental learning from positive data, J. Comput. Syst. Sci. 53 (1996) 88–103.
[18] J. Royer, J. Case, Subrecursive Programming Systems: Complexity and Succinctness, Research Monograph, in: Progress in Theoretical Computer Science, Birkhäuser, Boston, 1994.
[19] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw–Hill, New York, 1967, Reprinted by MIT Press, Cambridge, MA, 1987.

[20] G. Schäfer-Richter, Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien, PhD thesis, RWTH Aachen, 1984.
[21] K. Wexler, P. Culicover, Formal Principles of Language Acquisition, MIT Press, Cambridge, MA, 1980.
[22] R. Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, Elektronische Informationverarbeitung und Kybernetik 12 (1976) 93–99.