Iterative Learning from Positive Data and Counters

Timo Kötzing[⋆,⋆⋆]

Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany
koetzing@mpi-inf.mpg.de

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 for understanding what properties of a counter can benefit learning.

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, and that, for iterative learning criteria with one of these types of counter, separations of learning criteria are necessarily witnessed by classes containing only infinite languages.

Keywords: Inductive Inference.

1 Introduction

We analyze the problem of algorithmically learning a description for a formal language (a computably enumerable subset of the natural numbers) when presented successively all and only the elements of that language. For example, a learner h might be presented 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 no even number not divisible by 4 has been presented. 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.

[⋆] Timo Kötzing was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. NE 1182/5-1.

[⋆⋆] The author would like to thank John Case, Sanjay Jain, Frank Stephan and Sandra Zilles for valuable and fruitful discussions.

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 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 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 text book [JORS99]).

In this paper we are concerned with a memory limited variant of TxtEx-learning, namely *iterative learning* [Wie76, LZ96] (**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 for learning strictly fewer classes of languages. The successive literature analyzed iterative learners with some additional resources, for example a *bounded example memory* [LZ96]; "long term" *finite memory states* [FKS95]; or *feedback learning*, i.e. the ability to ask for the containment of examples in previously seen data [LZ96, CJLZ99].

A different option for providing additional learning power for iterative learning was suggested in [CM08b], 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). [CM08b] shows that this learning criterion is strictly more powerful than plain iterative learning, strictly less powerful than TxtEx-learning, and incomparable to *set-driven* learning [WC80].

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 introduces in [CM08b]). It is easy to see that this learning criterion is equivalent to *partially set-driven* (or *rearrangement independent*) learning [SR84]; it is well known that partially set-driven learning is equivalent to TxtEx-lerning.

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?

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

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 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: $\vec{\mathrm{R}}^{!} = \{c \mid \forall i : c(i+1) > c(i)\}$;
(iii) Monotone & unbounded: $\vec{\mathrm{R}} = \{c \mid \forall i : c(i+1) \geq c(i) \ \wedge\ \liminf_{i\to\infty} c(i) = \infty\}$;
(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 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).

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

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

[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 $\mathbb{N}$.

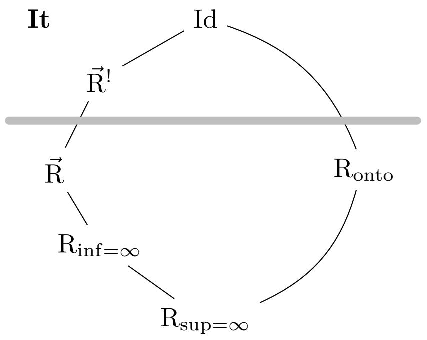
indefinitely, but only the behavior on large counter values affects the learning process in the limit.

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 these stages.

Counters from $\vec{\mathrm{R}}^!$ 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 $\mathrm{R}_{\mathrm{onto}}$ 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, $\mathrm{R}_{\mathrm{sup}=\infty}$, to plain iterative learning allows for an increase in learning power; however, there is no increase on learning classes of infinite languages only (Theorem 4). 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).


It
Id
$\vec{\mathrm{R}}^!$
$\vec{\mathrm{R}}$
$\mathrm{R}_{\mathrm{onto}}$
$\mathrm{R}_{\mathrm{inf}=\infty}$
$\mathrm{R}_{\mathrm{sup}=\infty}$

In particular, only the strict monotonicity of a counter gives additional learning power over $\mathrm{R}_{\mathrm{sup}=\infty}$ counters. The proofs for the claims inherent in the diagram can be found in Section 5.

Theorem 9 in Section 5 shows the separation depicted in the above diagram; its proof uses a self-learning class of languages [CK10, CK11] and Case's *Operator Recursion Theorem* (ORT) [Cas74, JORS99]. Because of space limitations, we only sketch that argument below.

Extending these results to settings where learners have additional resources is ongoing work; preliminary results show that, for 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 2 we show and discuss that this is not the case.

After this, the paper is organized as follows. Section 3 gives some mathematical preliminaries. In Section 4 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 5 gives some details for the diagram above.

2 Differences in Counters

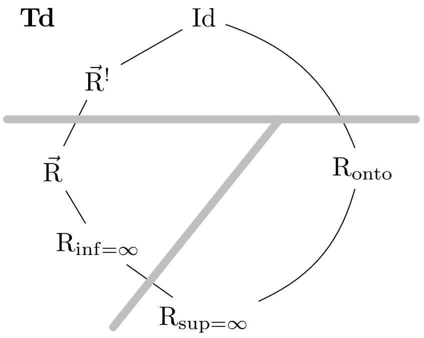
In this section we show that, for any choice of two different counter types, there is a learning criterion which, when augmented with one of the counter types, yields different classes of languages learnable than when augmented with the other.

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 [LZ96, CJLZ99]). 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 $\mathrm{R}_{\mathrm{onto}}$ counters which are not learnable with $\vec{\mathrm{R}}^!$ counters. Thus, there are settings where Id separates from $\vec{\mathrm{R}}^!$, and where $\mathrm{R}_{\mathrm{onto}}$ separates from $\mathrm{R}_{\mathrm{sup}=\infty}$.

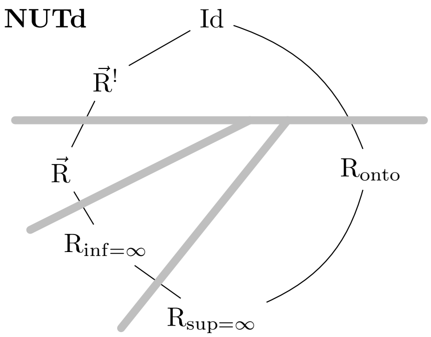
For more separations, we turn to very simple learning criteria. We consider *transductive* learning (**Td**), that is, learning without memory (which equals a degenerate case of memoryless learning with bounded memory states, where the bound on the number of states is 1; [CCJS07, CK08]). In this somewhat artificial toy setting a learner is presented 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.

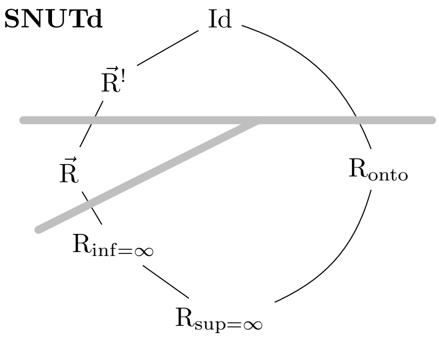
It is not hard to see that, for transductive learning, adding an $\mathrm{R}_{\mathrm{sup}=\infty}$ or $\mathrm{R}_{\mathrm{onto}}$ counter does not improve learning power. However, other types of counter do provide increases. The general result is depicted in the following diagram, using the same format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


Td
Id
$\vec{\mathrm{R}}^!$
$\vec{\mathrm{R}}$
$\mathrm{R}_{\mathrm{onto}}$
$\mathrm{R}_{\mathrm{inf}=\infty}$
$\mathrm{R}_{\mathrm{sup}=\infty}$

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 [BCM$^+$08]) or not even syntactically (strongly non-U-shaped learning [CM08a]). The resulting groupings and separations are depicted in the following two diagrams.


NUTd
Id
$\vec{\mathrm{R}}!$
$\vec{\mathrm{R}}$
$\mathrm{R}_{\mathrm{onto}}$
$\mathrm{R}_{\mathrm{inf}=\infty}$
$\mathrm{R}_{\mathrm{sup}=\infty}$


SNUTd
Id
$\vec{\mathrm{R}}!$
$\vec{\mathrm{R}}$
$\mathrm{R}_{\mathrm{onto}}$
$\mathrm{R}_{\mathrm{inf}=\infty}$
$\mathrm{R}_{\mathrm{sup}=\infty}$

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 $\mathrm{R}_{\mathrm{inf}=\infty}$ and $\vec{\mathrm{R}}$. For strongly non-U-shaped learning with $\mathrm{R}_{\mathrm{inf}=\infty}$ 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.

Because of space restrictions, the only theorem regarding transductive learning we will prove in this paper is given in Theorem 10, giving a flavor of the proofs concerning transductive learning.

3 Mathematical Preliminaries

Unintroduced notation follows [Rog67].

$\mathbb{N}$ 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 $f \in \mathfrak{P}$ *converges to* p, i.e., $\exists x_0 : f(x_0) = p \wedge \forall x \geq x_0 : f(x)\downarrow \in \{?, p\}$.[5]

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

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 $\mathrm{content}(q) = (\mathrm{range}(q) \setminus \{\#\})$.

3.1 Learning Criteria

A *learner* is a partial computable function.

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.

For any type of counters 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 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/counter to produce a sequence of conjectures.

[4] For a linear-time example, see [RC94, Section 2.3].

[5] $f(x)$ converges should not be confused with f converges *to*.

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

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

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

Thus, in iterative learning, the learner has access to the previous conjecture, but not so in transductive learning.

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 Sections 1 and 2. Formally, we let, for all p, T,

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

We combine any two sequence acceptance criteria δ and δ' by intersecting them.

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

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

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

4 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 $\Theta(\mathcal{L})$ (element wise application of Θ).[7]

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.

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

[7] [CK11] explores some notions of robustness for function learning.

Theorem 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 $\mathbf{TxtCtr}[R]\mathbf{ItEx}$ is Θ_0-robust.

Proof. Let $\mathcal{L} \in \mathbf{TxtCtr}[R]\mathbf{ItEx}$. 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 $\leq z$ 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 $\leq z + 1$. □

Theorem 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. □

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

Corollary 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 $\mathbf{TxtCtr}[R]\mathbf{ItEx}$ and $\mathbf{TxtCtr}[R']\mathbf{ItEx}$ 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.

5 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 (Theorem 4). The Theorems 5 through 8 give equivalences of learning power as indicated in Section 1. Finally, Theorem 9 gives the separation between strictly monotone counters and weaker counters.

Looking into the proof of Theorem 4 in [CM08b] (showing that an Id counter allows for learning languages which cannot be learned set-drivenly), we see that even the counters from $\mathrm{R}_{\mathrm{sup}=\infty}$ allow for learning more than learning set-drivenly. This leads to the first part of the next theorem. However, this proof makes use of finite languages. John Case remarked that $\mathrm{R}_{\mathrm{sup}=\infty}$-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* `ce` sets.

Theorem 4. We have

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

and

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

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

Theorem 5. We have

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

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, without loss of generality, the new counter is from $\mathrm{R}_{\mathrm{inf}=\infty}$. 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 $\mathcal{L} \in \mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{ItEx}$ as witnessed by h_0. By Corollary 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

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

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 $c \in \mathrm{R}_{\mathrm{sup}=\infty}$ 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 $\mathrm{R}_{\mathrm{sup}=\infty}$; this converges, a contradiction.

Suppose, by way of contradiction, h on T and c does not converge 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 $s' \geq s$, $h_0(e, \#, s') \neq e$, that is, h_0 changes its mind on $\#$ for all but finitely many counter values. Then, clearly, at some point after the convergence of h on T, we get a counter value $\geq s$ 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, $\max_t[h_0(e, T'(i), t) = e] \leq s$. Because of the case we consider now, we know that there are infinitely many i with $T'(i) \neq \#$ and $\max_t[h_0(e, T'(i), t) = e] \leq s$. 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 6. 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, z, we let $f(e, x, z)$ be the least t with $e \leq t \leq \max(e, z)$ 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 with some t, then $c'(i) = t$. Otherwise, if there will be a mind change of h on T with counter c later, then $c'(i) = e$, else $c'(i) = \max(e, \min\{c(j) \mid j \geq i\})$.

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

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 7. 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, we get that a class of languages all containing all odd data witness the separation as well. From a text for such a languages we can extract a complete counter (by dividing each datum by 2, rounding down), a contradiction. □

Last we show that also the top-most in the counter hierarchy gives no difference in learning power.

Theorem 8. 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,

$$\begin{aligned} h_0^*(e, \emptyset, z) &= e; \\ \forall \sigma, x : h_0^*(e, \sigma\, x, z) &= h_0(h_0^*(e, \sigma, z), x, z + \mathrm{len}(\sigma)). \end{aligned}$$

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

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

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

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

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. □

The next theorem shows that the remaining two classes of learning power do separate.

Theorem 9. We have

$$\mathbf{Txt}\,\mathbf{Ctr}[\vec{\mathrm{R}}]\mathbf{ItEx} \subset \mathbf{Txt}\,\mathbf{Ctr}[\vec{\mathrm{R}}^{!}]\mathbf{ItEx}.$$

Informal sketch of the argument. 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 [CK10, CK11]). We will then suppose that this class can be learned with $\vec{\mathrm{R}}$ counters by some function h. We will suggest to h to change its mind on certain data (we call them $a(0)$, $a(1)$, $\ldots$); if h never changes its mind, then this will suffice to make h fail. Otherwise, we are still free to suggest to h not to change its mind on this data for high counter values (above some threshold we call c_0). We now add some other data (we call them $b(0)$, $b(1)$, $\ldots$) on which h should not change, unless it has changed on some a-values previously. In fact, we add exactly c_0 such data points. With a counter from $\vec{\mathrm{R}}$, these data points may all come very early, on a low and always the same counter value; h will not be able to change its mind then (otherwise we just start over and find infinitely many mind changes on data where no mind change was suggested). Furthermore, h will change its mind later on some a, leading to no success in identification.

With a strictly increasing counter, however, h would have no problem: if all b-values come before the a-values, then the counter would be pushed high enough such that on a-values there need not be a mind change. □

Finally, we give two theorems regarding transductive learning.

Theorem 10. We have

$$\mathbf{Txt}\,\mathbf{Ctr}[\mathrm{R}_{\mathrm{onto}}]\mathbf{TdEx} = \mathbf{Txt}\,\mathbf{TdEx} = \mathbf{Txt}\,\mathbf{Ctr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{SNUTdEx}.$$

Proof. We start with the first equality, where the inclusion "$\supseteq$" is trivial. Let $\mathcal{L}$ be $\mathbf{Txt}\,\mathbf{Ctr}[\mathrm{R}_{\mathrm{onto}}]\mathbf{TdEx}$-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 infinitely output on learning any language, in which case $\mathcal{L}$ can have at most one element, which we can learn using only the initial conjecture and outputting ? on # with any counter value.

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

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

Suppose, by way of contradiction, that we can get two different outputs p and p' on two datum/counter pairs. Then we can force infinite oscillation between p and p', a contradiction. Thus, there is at most one p ever output on a datum. Suppose that, for each $x \in L$, there is a z such that $h(x, z) =?$. Then we can list all $x \in L$ such that h always outputs ? and fill up missing counter values with $\#$. As L is learned by h, $L = W_{h(\emptyset)}$.

Clearly, any h as above might as well ignore the counter and learn without such additional help.

Regarding $\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{SNUTdEx}$, it is easy to see that it includes all $\mathbf{TxtTdEx}$-learnable classes (using the characterization of learnability as given by the above list). Furthermore, note that any two syntactically different outputs of a learner lead to a syntactic U-shape on some text/counter, with the counter from $\mathrm{R}_{\mathrm{inf}=\infty}$. Thus, the above characterization for languages from $\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{onto}}]\mathbf{TdEx}$ also characterizes the languages from that are learnable in the sense of $\mathbf{TxtCtr}[\mathrm{R}_{\mathrm{inf}=\infty}]\mathbf{SNUTdEx}$. □

Theorem 11. 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 $\langle T, c\rangle$ makes the same infinitely many non-? outputs, and all of those are equal and correct.

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. □

References

[BCM+08] Baliga, G., Case, J., Merkle, W., Stephan, F., Wiehagen, W.: When unlearning helps. Information and Computation 206, 694–709 (2008)

[Cas74] Case, J.: Periodicity in generations of automata. Mathematical Systems Theory 8, 15–32 (1974)

[CCJS07] Carlucci, L., Case, J., Jain, S., Stephan, F.: Results on memory-limited U-shaped learning. Information and Computation 205, 1551–1573 (2007)

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

[CK08] Case, J., Kötzing, T.: Dynamic modeling in inductive inference. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) ALT 2008. LNCS (LNAI), vol. 5254, pp. 404–418. Springer, Heidelberg (2008)

[CK10] Case, J., Kötzing, T.: Strongly non-U-shaped learning results by general techniques. In: Proceedings of COLT (Conference on Learning Theory), pp. 181–193 (2010)

[CK11] Case, J., Kötzing, T.: Measuring learning complexity with criteria epitomizers. In: Proceedings of STACS (Symposion on Theoretical Aspects of Computer Science), pp. 320–331 (2011)

[CM08a] Case, J., Moelius, S.: Optimal language learning. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) ALT 2008. LNCS (LNAI), vol. 5254, pp. 419–433. Springer, Heidelberg (2008)

[CM08b] Case, J., Moelius, S.: U-shaped, iterative, and iterative-with-counter learning. Machine Learning 72, 63–88 (2008)

[FKS95] Freivalds, R., Kinber, E., Smith, C.: On the impact of forgetting on learning machines. Journal of the ACM 42, 1146–1168 (1995)

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

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

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

[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)

[SR84] Schäfer-Richter, G.: Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien. PhD thesis, RWTH Aachen (1984)

[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)