Information and Computation 251 (2016) 1–15

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Strongly non-U-shaped language learning results by general techniques

John Case [a], Timo Kötzing [b,*]

[a] Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA
[b] Hasso Plattner Institute, 14482 Potsdam, Germany

ARTICLE INFO

Article history:
Received 24 February 2014
Received in revised form 8 June 2015
Available online 1 July 2016

Keywords:
Inductive inference
Non-U-shaped learning
General techniques
Self-learning classes
Infinitary self-referential programs

ABSTRACT

In learning, a semantic or behavioral U-shape occurs when a learner first learns, then unlearns, and, finally, relearns, some target concept.
This paper introduces two general techniques and applies them especially to syntactic U-shapes in learning: one technique to show when they are necessary and one to show when they are unnecessary. The technique for the former is very general and applicable to a much wider range of learning criteria. It employs so-called *self-learning classes of languages* which are shown to *characterize* completely one criterion learning more than another.
We apply these techniques to show that, for set-driven and rearrangement-independent learning, any kind of U-shapes is unnecessary. Furthermore, we show that U-shapes *are* necessary *in a strong way* for iterative learning, contrasting with an earlier result by Case and Moelius that semantic U-shapes are *un*necessary for iterative learning.

1. Introduction

In Section 1.1 we explain U-shaped learning. In Section 1.2 we briefly discuss the general techniques of the present paper and summarize in Section 1.3 our applications of these techniques regarding the *necessity* of U-shaped learning.

1.1. U-shaped learning

U-shaped learning occurs when a learner first learns a correct *behavior*, then abandons that correct behavior and finally returns to it once again. This pattern of learning has been observed by cognitive and developmental psychologists in a variety of child development phenomena, such as language learning [6,31,38], understanding of temperature [38,39], weight conservation [5,38], object permanence [5,38] and face recognition [7]. The case of language acquisition is paradigmatic. For example, a child first uses *spoke*, the correct past tense of the irregular verb *to speak*. Then the child ostensibly overregularizes incorrectly using *speaked*. Lastly and finally the child returns to using *spoke*. The language acquisition case of U-shaped learning behavior has figured prominently in cognitive science [31,34,41].

While the prior cognitive science literature on U-shaped learning was typically concerned with modeling *how* humans achieve U-shaped behavior, the papers [3,11] are motivated by the question of *why* humans exhibit this seemingly inefficient behavior. Is it a mere harmless evolutionary inefficiency or is it *necessary* for full human learning power? A technically

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

http://dx.doi.org/10.1016/j.ic.2016.06.015
0890-5401/

answerable version of this question is: are there some formal learning classes of tasks for which U-shaped behavior is logically necessary? We first need to describe some formal criteria of successful learning.

An algorithmic learning function h is, in effect, fed an infinite sequence consisting of the elements of a (formal) language L in arbitrary order with possibly some pause symbols # in between elements. During this process, h outputs a corresponding sequence $p(0), p(1), \ldots$ of hypotheses (grammars) which may generate the language L to be learned. A fundamental criterion of successful learning of a language is called *explanatory learning* (TxtEx-*learning*, also called *learning in the limit*) and was introduced by Gold [27]. Explanatory learning requires that the learner's output conjectures stabilize in the limit to a *single* conjecture (grammar/program, description/explanation) that generates the input language. *Behaviorally correct learning* [18,33] requires, for successful learning, convergence in the limit to a *sequence* of correct (but possibly syntactically distinct) conjectures. Another interesting class of criteria features *vacillatory learning* [10,28]. This paradigm involves learning criteria which allow the learner to vacillate in the limit between *at most* some bounded, finite number of syntactically distinct but correct conjectures. For each criterion that we consider above (and below), a *non-U-shaped learner* is naturally modeled as a learner that never returns to a previously *semantically* abandoned correct conjecture on languages it learns according to that criterion.

Ref. [3] showed that every TxtEx-learnable class of languages is TxtEx-learnable by a non-U-shaped learner, that is, for TxtEx-learnability, U-shaped learning is *not* necessary. Furthermore, based on a proof in [24,3] noted that, by contrast, for behaviorally correct learning [23,1,18,33], U-shaped learning *is* necessary for full learning power. In [11] it is shown that, for non-trivial vacillatory learning, U-shaped learning is again necessary (for full learning power). Thus, in many contexts, seemingly inefficient U-shaped learning can actually increase one's learning power.

What turns out to be a variant of non-U-shaped learning is *strongly non-U-shaped* learning essentially defined in [43],[1] where the learner is required never to *syntactically* abandon a correct conjecture on languages it learns according to that criterion. Clearly, *strong* non-U-shaped learnability implies non-U-shaped learnability.[2] In our experience, for theoretical purposes, it is frequently easier to show non-U-shaped learnability by showing *strong* non-U-shaped learnability. Herein we especially study strong non-U-shaped learnability.

1.2. Presented techniques

The present paper presents two general techniques to tackle problems regarding U-shaped learning.

The first general technique can be used to show the *necessity* of U-shapes and employs so-called *self-learning classes of languages*. These are explained in Section 3 below. These self-learning classes of languages provide a (provably) most general way for finding classes of languages that separate two learning criteria, i.e., they give a general way of finding an *example* class of languages learnable with a given learning criterion, but not with another. Theorem 3.6 implies that its presented self-learning classes *necessarily* separate two learnability sets — *iff any class does*. This technique is not specialized only to analyze U-shaped learning, but can be applied to other learning criteria as well. The technique is developed and discussed further in Section 3.

The second general technique is used to show that syntactic U-shapes are *un*necessary and is phrased as a characterization of strongly non-U-shaped learnability of classes of languages (Theorem 4.4).

1.3. Applications of general techniques

A learning machine is *set-driven* [42,37,26,28] (respectively, *rearrangement-independent* [37,26,28]) iff, at any time, its output conjecture depends only on the *set* of non-pause data it has seen (respectively, set of non-pause data *and* data-sequence length), *not* on the *order* of that data's presentation. Child language learning may be insensitive to the order or timing of data presentation; set-drivenness and rearrangement independence, **Sd** and **Ri**, respectively, provide two *local* notions of such insensitivity [10]. It is interesting, then, to see the interaction of these notions with forbidding U-shapes of one kind or another. As we shall see in Section 5, Theorems 5.2 and 5.3, proved with the aid of a general technique from Section 4, imply, for these local data order insensitivity notions, for TxtEx-learning, U-shapes, *even in the strong sense* are *un*necessary.

An *iterative* learner outputs its conjectures only on the basis of its immediately prior conjecture (if any) and its current datum. As we shall see in Section 5, iterative learning provides a (first) example of a setting in which non-U-shaped and strongly non-U-shaped learning are extensionally distinct: [19] shows semantic U-shapes to be *un*necessary for iterative learning, while Theorem 5.7 in the present paper implies that they are *in the strong sense* necessary. To prove this latter result, we actually modify the self-learning class of languages from Theorem 3.6 to make it easier to work with — although the original version *must* work too (by Theorems 3.6 and 5.7).[3]

[1] Wiehagen actually used the term *semantically finite* in place of *strongly non-U-shaped*. However, there is a clear connection between this notion and that of *non-U-shapedness*. Our choice of terminology is meant to expose this connection. See also [21].

[2] For non-U-shaped learning, the learner (on the way to success) must not *semantically* abandon a correct conjecture. In general, semantic change of conjecture is not algorithmically detectable, but syntactic change is. *However*, in the cognitive science lab we can many times see a *behavioral/semantic* change, but it is beyond the current state of the art to see, for example, grammars in people's heads — so we can't *yet* see mere syntactic changes in people's heads.

[3] Some recent papers [13,15,16,30] have also employed (different) self-learning classes for separations.

Some of our proofs involve subtle infinitary program self-reference arguments employing (variants of) the Case's Operator Recursion Theorem (ORT) from [8,9,28,32].

Note that the present paper is an extended version of [14].

1.4. Open problems

Some problems regarding the necessity of U-shapes of one kind or another still remain open. An *iterative with counter learner* is an iterative learner which, in making a conjecture, also has access to the data-sequence *length* so far. For example, it is still open whether semantic U-shapes are necessary for iterative with counter learning — as asked in [19]. If so, then the relevant self-learning class from Theorem 3.6 below *must* provide a separation.

See the end of Section 5 for some more open problems regarding the necessity of any one of the two kinds of U-shapes for learning criteria of the present paper.

2. Mathematical preliminaries

Unintroduced complexity theoretic notation follows [35]. Other unintroduced notation follows [36].

By $\mathbb{N}$ we denote the set of natural numbers, $\{0, 1, 2, \ldots\}$. We let $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. The symbols $\subseteq$, $\subset$, $\supseteq$, $\supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets. For any set A, we let $\mathrm{Pow}(A)$ denote the set of all subsets of A. We let Δ denote the symmetric difference of two sets. $\emptyset$ denotes both the empty set and the empty sequence. $\mathfrak{P}$ denotes the set of all partial functions $\mathbb{N} \to \mathbb{N}$.

The quantifier $\forall^\infty x$ means "for all but finitely many x", the quantifier $\exists^\infty x$ means "for infinitely many x".

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

Whenever we consider tuples of natural numbers as input to a function, it is understood that they are suitably encoded as a single natural number; similarly for finite sets and sequences, see, for example, [35] for details.

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"); note that typically the learner can remember its past conjecture and is thus able to output this past hypothesis instead of ?. We write $f \to p$ to denote that $f : \mathbb{N} \to \mathbb{N} \cup \{?\}$ *converges to* p, i.e., $\exists x_0 : f(x_0) = p \wedge \forall x \geq x_0 : f(x)\downarrow \in \{?, p\}$.[4] $\mathcal{P}$ and $\mathcal{R}$ denote, respectively, the set of all partial computable and the set of all computable functions (mapping $\mathbb{N} \to \mathbb{N}$).

We let φ be any fixed acceptable programming system for $\mathcal{P}$ ([36], an acceptable programming system could, for example, be based on a natural programming language such as C or Java, or on Turing machines). Further, we let φ_p denote the partial computable function computed by the φ-program with code number p. A set $L \subseteq \mathbb{N}$ is *computably enumerable* (ce) iff it is the domain of a partial computable function. Let $\mathcal{E}$ denote the set of all ce sets. We call a set *computable* iff its characteristic function is computable. We let W be the mapping such that $\forall e : W(e) = \mathrm{dom}(\varphi_e)$. For each e, we write W_e instead of $W(e)$. W is, then, a mapping from $\mathbb{N}$ *onto* $\mathcal{E}$. We say that e is an index, or program, (in W) for W_e.

The symbol # is pronounced *pause* and is used to symbolize "no new input data". 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 $f \in \mathfrak{P}$ and all i, we use $f[i]$ to denote the sequence $f(0), \ldots, f(i-1)$ (the empty sequence if $i = 0$ and undefined, if one of these values is undefined).

We will make use of a padded variant of the s-m-n Theorem [36]. Intuitively, s-m-n permits algorithmic storage of arbitrary data (and, hence, programs) inside any program. The suitable padded variant of s-m-n we use herein states that there is a strictly monotonic increasing computable function s such that

$$\forall a, b, c : \varphi_{s(a,b)}(c) = \varphi_a(b, c). \tag{1}$$

In (1), φ-program $s(a, b)$ is essentially φ-program a with datum b stored inside. We will also use a suitably padded version of Case's *Operator Recursion Theorem* (**ORT**), providing *infinitary* self (and other) reference [8,9,28]. See [36] for a treatment of computable operators (also called recursive operators). **ORT** itself states that, for all computable operators $\Theta : \mathfrak{P} \to \mathfrak{P}$,

$$\exists e \in \mathcal{R} \forall a, b : \varphi_{e(a)}(b) = \Theta(e)(a, b).$$

In the padded version we employ, the function e will also be strictly monotone increasing.

2.1. Computability-theoretic learning

In this section we formally define several criteria for computability-theoretic learning.

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

In this section we formally introduce our setting of learning in the limit and associated learning criteria. We follow [29] in its "building-blocks" approach for defining learning criteria.

A *learner* is a partial function from $\mathbb{N}$ to $\mathbb{N} \cup \{?\}$ (note that, for this paper, we do not always require computability of learners). A *language* is a $\subseteq$ set $L \subseteq \mathbb{N}$. Any total function $T : \mathbb{N} \to \mathbb{N} \cup \{\#\}$ is called a *text*. Note that the only text for the empty language is an infinite sequence of #. For any given language L, a *text for* L is a text T such that $\text{content}(T) = L$. This kind of text is what learners usually get as information; we assume that the #-symbol and the data are appropriately coded to serve as input to (partial) computable functions. With $\mathbf{Txt}(L)$ we denote the set of all texts for L.

A *sequence generating operator* is an operator β taking as arguments a function h (the learner) and a text T and outputting a one-argument function. For p the output of β on h and T we call p the *learning sequence* of h given T. Intuitively, β defines how a learner can interact with a given text to produce a sequence p of conjectures.

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

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

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

Thus, in iterative learning, the learner has access to the previous conjecture and the current datum, but not directly to any strictly previous data as in **G**-learning.

Another interesting sequence generating operator is set-driven learning ([42], denoted **Sd**). We let, for all learners h, texts T and all i,

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

A slight variant of set-driven learning, called *rearrangement-independent learning* additionally provides information about the length of the input sequence. We let, for all learners h, texts T and all i,

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

Finally, we are also interested in iterative-with-counter learning [19]; for all learners h, texts T and all i, we let

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

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

A *sequence acceptance criterion* is a predicate δ on a learning sequence and a text. We give the examples of explanatory (**Ex**) non-U-shaped (**NU**) and strongly non-U-shaped (**SNU**) learning, which were discussed in Section 1. Formally, we define, for all conjecture sequences p and texts T,

$$\mathbf{Ex}(p,T) \Leftrightarrow [\exists q : \text{content}(T) = W_q \wedge \forall^{\infty} i : p(i) = q];$$

$$\mathbf{NU}(p,T) \Leftrightarrow [\forall i,j,k : i \le j \le k \wedge W_{p(i)} = W_{p(k)} = \text{content}(T) \Rightarrow W_{p(i)} = W_{p(j)}];$$

$$\mathbf{SNU}(p,T) \Leftrightarrow [\forall i,j,k : i \le j \le k \wedge W_{p(i)} = W_{p(k)} = \text{content}(T) \Rightarrow p(i) = p(j)].$$

We combine any two sequence acceptance criteria δ and δ' by intersecting them; we denote this by juxtaposition (for example, **NU** is meant to be always used together with **Ex**).

For any set of allowed learners $\mathcal{C} \subseteq \mathcal{P}$, sequence generating operator β and any sequence acceptance restriction δ, $(\mathcal{C}, \beta, \delta)$ is a *learning criterion*. We also write this learning criterion as $\mathcal{C}\mathbf{Txt}\beta\delta$ and omit $\mathcal{C}$ if $\mathcal{C} = \mathcal{P}$. A learner h $\mathcal{C}\mathbf{Txt}\beta\delta$*-learns* the empty set if $h \notin \mathcal{C}$ and otherwise the set

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

With $[\mathcal{C}\mathbf{Txt}\beta\delta]$ we denote the collection of all sets of language $\mathcal{L}$ such that there is a learner h with $\mathcal{L} \subseteq \mathcal{C}\mathbf{Txt}\beta\delta(h)$.

With these definitions we see that the informally introduced criterion TxtEx from the introduction corresponds to **TxtGEx**.

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

Starred Learners
For two sequence generating operators β and β' we write $\beta \preceq \beta'$ if, for all h there is h' with

$$\forall T : \beta(h, T) = \beta'(h', T).$$

Intuitively, any learners learning with information corresponding to β can be turned into learners learning according to information from β'. Clearly, for all β give above, we have $\beta \preceq \mathbf{G}$. Thus, for any β-learner h, we let h^* be the corresponding **G**-learner. That is, for all sequences σ, $h^*(\sigma)$ denotes the current conjecture of h after being fed the sequence σ of data items.

In particular, for $h \in \mathcal{P}$ and σ a sequence, we have the following.

- If h is a set-driven learner:
$$h^*(\sigma) = h(\text{content}(\sigma)). \tag{2}$$
- If h is a rearrangement-independent learner:
$$h^*(\sigma) = h(\text{content}(\sigma), \text{len}(\sigma)). \tag{3}$$

2.2. Locking

In this section we will make a number of definitions in dependence on a learning criterion $\mathcal{C}\mathbf{Txt}\beta\delta$ and a learner $h \in \mathcal{C}$.

We first define the concept of a stabilizer sequence (introduced in [2] and called "stabilizer segment" in [26]). Let β be a sequence generating operator. Let L be a language and $h \in \mathcal{P}$ a learner. A sequence σ with elements from L is said to be a *β-stabilizer sequence of h on L* iff

$$(\forall T \in \mathbf{Txt}(L) | \sigma \subseteq T) \forall i \geq \text{len}(\sigma) : \beta(h, T)(\text{len}(\sigma)) = \beta(h, T)(i);$$

Intuitively, a stabilizer sequence σ of h on L is a sequence of elements from L such that h on any text extending σ will never make a change of conjecture after having seen σ. For any stabilizer sequence σ of h on L, we call σ a *locking sequence* if the conjecture of h after σ is a correct conjecture.

It is well known that, if a learner h **TxtGEx**-learns a language L, then there is a stabilizer sequence of h on L (see [28]). However, texts do not necessarily contain such a sequence as an initial segment. Below, we define a learning restriction that requires a learner and a language to have stabilizer sequences as initial sequences of *all* texts for the language. This motivates us to define the following notion. We say that a learner h $\mathcal{C}\mathbf{Txt}\beta\delta$-learns a set of languages $\mathcal{L}$ *stabilizingly* iff h $\mathcal{C}\mathbf{Txt}\beta\delta$-learns $\mathcal{L}$ and, for all $L \in \mathcal{L}$ and T a text for L, there is n such that $T[n]$ is a β-stabilizer sequence of h on L.

We define a *β-sink of h on L* to be a conjecture e such that

$$\forall T \in \mathbf{Txt}(L) \forall i_0 : [\beta(h, T)(i_0) = e \Rightarrow (\forall i \geq i_0)(\beta(h, T)(i) = e)].$$

Intuitively, a sink is a conjecture never abandoned on texts for L. A stabilizer sequence σ for h on a language L is called a *sink-stabilizer sequence for h* on L iff for all texts T for L with $\sigma \subseteq T$ we have that $\beta(h, T)(\text{len}(\sigma))$ is a β-sink of h on L.

We say that a learner h $\mathcal{C}\mathbf{Txt}\beta\delta$-learns a set of languages $\mathcal{L}$ *sink-stabilizingly* iff h $\mathcal{C}\mathbf{Txt}\beta\delta$-learns $\mathcal{L}$ and, for all $L \in \mathcal{L}$ and T a text for L, there is n such that $T[n]$ is a β-sink-stabilizer sequence of h on L.

Sink-stabilizing is of interest, as we show a characterization theorem (Theorem 4.4 below) of strongly non-U-shaped learning in terms of sink-stabilizing learning.

3. Self-learning classes of languages for separations

In this section we discuss a way of showing U-shapes to be *necessary*. Formally, this is done via showing that a learnability class *separates* from its non-U-shaped variant.

The approach described below is very general and can be applied to show separation results in many other areas of computability-theoretic learning in the limit as well.

The key idea is that of *self-learning classes of languages*. In the previous literature, self-*describing* classes of languages have been used.[6] A particularly simple example class of self-describing languages, taken from [18, Theorem 1], is

$$\mathcal{L}_0 = \{L \text{ is computable} \mid L \neq \emptyset \wedge W_{\min L} = L\}. \tag{4}$$

Intuitively, each $L \in \mathcal{L}_0$ gives a complete *description* of itself, encoded (as a W-index) within only finitely many (in fact, one) of its elements. It is well-known, using standard computability theoretic arguments, that these kind of classes of languages are very big (for example, $\mathcal{L}_0$ contains a finite variant[7] of any given computable language, which can easily be seen using Kleene's Recursion Theorem).

[6] See [28]. In there, the term "self-describing" was used on page 71 in the context of function learning and extended on page 97 to language learning.
[7] A set A is a finite variant of a set B iff $A \Delta B$ is a finite set.

Many variants of self-describing classes of languages have been used for separation results within computability-theoretic learning (see, for example, [2,18,22,10,28]). Showing a separation with a complicated self-describing class of languages sometimes requires a non-trivial learner (see, for extreme examples, [12]).

We now take the technique of self-describing one step further. A self-*learning* class of languages is such that each element of each language of the class provides instructions for what to compute and output as a new hypothesis. Thus, all a learner needs to do is to execute the instructions given by its latest datum. For example, an informal[8] learner h_1 can be defined such that, for all sequences σ and numbers x,

$$h_1(\sigma \diamond x) = \varphi_x(\sigma \diamond x). \tag{5}$$

Intuitively, h_1 interprets the latest datum as a program in the φ-system and runs this program on all known data, which is $\sigma \diamond x$. Variants of this h_1 can be defined to obtain learners with special additional properties, such as totality or set-drivenness (see Theorem 3.6).

In practice, the general scheme is as follows. Suppose we want to show, for two learning criteria I_0 and I_1, $[I_1] \setminus [I_0] \neq \emptyset$. Then we define a simple learner, for example h_1 above, and let $\mathcal{L}_1$ be the class of all languages I_1-learned by h_1. All that would remain to be shown is that $\mathcal{L}_1$ is not I_0-learnable, which can often be done using **ORT**.

Below, in Theorem 3.6, we give a *very general* result regarding some self-learning classes of languages *guaranteed* to witness separations when they exist. In order to do so, we proceed next by making some formal definitions.

For a function $e \in \mathcal{P}$ and a language L, we let $e(L) = \{e(x) \mid x \in L\}$; for a class of languages $\mathcal{L}$, we let $e(\mathcal{L}) = \{e(L) \mid L \in \mathcal{L}\}$.

Let $\mathcal{P}_{\text{c1-1}} \subseteq \mathcal{P}$ (respectively, $\mathcal{R}_{\text{c1-1}} \subseteq \mathcal{R}$), denote the set of all 1-1 partial (respectively, total) computable functions with computable domain and range.

Definition 3.1. A learning criterion I is called *computably* $\mathcal{P}_{\text{c1-1}}$*-invariant* iff there is a computable operator $\Theta : \mathfrak{P}^2 \to \mathfrak{P}$ such that

$$\forall h \in \mathcal{P}, \forall e \in \mathcal{P}_{\text{c1-1}} : e(I(h)) \subseteq I(\Theta(h, e)). \tag{6}$$

Intuitively, if a learner h learns languages L, then $\Theta(h, e)$ learns $e(L)$. Note that this property was called *robustness* in previous publications [15,30].

Remark 3.2. Let I be computably $\mathcal{P}_{\text{c1-1}}$-invariant. Then we have

$$\forall e \in \mathcal{P}_{\text{c1-1}}, \forall \mathcal{L} \subseteq \mathcal{E} : \mathcal{L} \in [I] \Leftrightarrow e(\mathcal{L}) \in [I].$$

Remark 3.3. For any of the choices $\mathcal{C} \in \{\mathcal{R}, \mathcal{P}\}$, $\beta \in \{\mathbf{G}, \mathbf{Ri}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}$ and $\delta \in \{\mathbf{SNUEx}, \mathbf{NUEx}, \mathbf{Ex}\}$ we have that $(\mathcal{C}, \beta, \delta)$ is computably $\mathcal{P}_{\text{c1-1}}$-invariant.

Proof. Let $\mathcal{L}$ be $\mathcal{C}\mathbf{Txt}\beta\delta$-learnable as witnessed by h and let $e \in \mathcal{P}_{\text{c1-1}}$. We show $e(\mathcal{L})$ is $\mathcal{C}\mathbf{Txt}\beta\delta$-learnable for the examples of $\beta \in \{\mathbf{G}, \mathbf{It}\}$, all other cases are similar. Using padded s-m-n, there is a strictly monotone increasing $f \in \mathcal{R}$ such that

$$\forall p : W_{f(p)} = e(W_p).$$

Regarding **G**: Let $h' \in \mathcal{P}$ be such that

$$\forall \sigma : h'(\sigma) = \begin{cases} f(h(e^{-1} \circ \sigma)), & \text{if content}(\sigma) \subseteq \text{range}(e); \\ 0, & \text{otherwise.} \end{cases}$$ [9]

Clearly, $e(\mathcal{L}) \subseteq \mathcal{C}\mathbf{TxtGEx}(h')$; it is easy to verify that (strongly) non-U-shapedness is preserved.

Regarding **It**: Let $h' \in \mathcal{P}$ be such that

$$\forall p, x : h'(p, x) = \begin{cases} f(h(f^{-1}(p), e^{-1}(x))), & \text{if } p \in \text{range}(f) \text{ and } x \in \text{range}(e); \\ 0, & \text{otherwise.} \end{cases}$$

The remaining reasoning for **It** is analogous to that for **G**. □

Definition 3.4. Let $I = (\mathcal{C}, \beta, \delta)$ be a learning criterion. We call I *data normal* iff, for all p_0 such that $W_{p_0} = \emptyset$, there is a computable operator $\hat{\Theta} : \mathfrak{P} \to \mathfrak{P}$ such that

$$I(h) \subseteq I(\hat{\Theta}(h)) \tag{7}$$

and (a)–(d) below.

[8] Note that we ignore the possible input of $x = \#$.

[9] Note that we use the convention $f(?) = ?$ and $e(\#) = \#$ for ease of notation.

(a) There is an $f_\beta \in \mathcal{R}$ such that, for all learners h, texts T and all $i > 0$,

$$\beta(h,T)(i) = h(f_\beta(T[i], \beta(h,T)[i])).^{10} \tag{8}$$

(b) There is a function $d_\beta \in \mathcal{R}$ such that, for all texts T and all i, if we have $\beta(\hat{\Theta}(h),T)[i]\downarrow$, then

$$d_\beta(f_\beta(T[i], \beta(\hat{\Theta}(h),T)[i])) \in \begin{cases} \{\#\}, & \text{if content}(T[i]) = \emptyset; \\ \text{content}(T[i]), & \text{otherwise.}^{11} \end{cases} \tag{9}$$

(c) For all $h \in \mathcal{P}$,

$$\forall \sigma, \tau : d_\beta(f_\beta(\sigma,\tau)) = \# \Rightarrow \hat{\Theta}(h)(f_\beta(\sigma,\tau)) = p_0.^{12} \tag{10}$$

(d) For all $h, h' \in \mathcal{P}$,

$$[\forall L \in I(h) \forall T \in \mathbf{Txt}(L) : \beta(h,T) = \beta(h',T)] \Rightarrow I(h) \subseteq I(h').^{13} \tag{11}$$

Remark 3.5. For any of the choices $\mathcal{C} \in \{\mathcal{R}, \mathcal{P}\}$, $\beta \in \{\mathbf{G}, \mathbf{Ri}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}$ and $\delta \in \{\mathbf{SNUEx}, \mathbf{NUEx}, \mathbf{Ex}\}$ we have that $(\mathcal{C}, \beta, \delta)$ is data normal.

Proof. Regarding (a), we can, for example, take, for each σ, τ,

$$f_{\mathbf{G}}(\sigma,\tau) = \sigma;$$
$$f_{\mathbf{Sd}}(\sigma,\tau) = \text{content}(\sigma);$$
$$f_{\mathbf{It}}(\sigma,\tau) = \begin{cases} \langle \text{last}(\tau), \text{last}(\sigma)\rangle, & \text{if } \sigma \neq \emptyset \text{ and } \tau \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases}$$

These work for each $\mathcal{C} \in \{\mathcal{R}, \mathcal{P}\}$. Similarly, we can get f_β for the other relevant β.

We show Requirements (b) and (c) for $\beta = \mathbf{It}$; all other cases are similar, but easier.

Let p_0 be such that $W_{p_0} = \emptyset$. By the padded s-m-n Theorem, there is a strictly monotone increasing $z \in \mathcal{R}$ such that $\forall e, a : W_{z(e,a)} = W_e$ and $p_0 \notin \text{range}(z)$. Then there is a $y \in \mathcal{R}$ such that $\forall e, a : y(z(e,a)) = a$. Let $\hat{\Theta}$ be such that

$$\forall h, e, x : \hat{\Theta}(h)(e,x) = \begin{cases} p_0, & \text{if } x = \# \text{ and } e \in \{?, p_0\}; \\ z(h(?,x),x), & \text{else if } e \notin \text{range}(z); \\ z(h(e',x),a), & \text{otherwise, with } e = z(e',a). \end{cases}$$

Let $d \in \mathcal{R}$ such that

$$\forall e, x : d(e,x) = \begin{cases} x, & \text{if } x \neq \#; \\ \#, & \text{else if } e \notin \text{range}(z); \\ a, & \text{otherwise, with } e = z(e',a). \end{cases}$$

Then d fulfills (b), and (c) is obvious.

Requirement (d) is clear for all criteria considered. □

Next (Theorem 3.6) is the main result of this section, giving *sufficient* conditions for when a separation will necessarily be witnessed by a *specific* self-learning class of languages. As a corollary, we get that *for each pair of learning criteria* with components from $\{\mathcal{R}, \mathcal{P}\}$, $\{\mathbf{G}, \mathbf{Ri}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}$ and $\{\mathbf{SNUEx}, \mathbf{NUEx}, \mathbf{Ex}\}$, any separations are witnessed by such classes! In this sense, self-learning classes of languages capture the *essence* of separations (when they exist). Note that the proof of the theorem would simplify a lot, were one to suppose somewhat stronger properties of the learning criteria, in particular, excluding the use of **It** and **ItCtr** as sequence generating operators. Theorem 3.6 can be modified to cover other kinds of criteria, for example, those pertaining to learnability by total learners. In a future paper, we will analyze self learning-classes in more depth and will provide further theorems like Theorem 3.6.

[10] Intuitively, the i-th conjecture of h on T depends only on some information (as specified by f_β) about the first i datapoints and conjectures.
[11] Intuitively, from the information given by f_β, a datum (if any) that this datum is based on can be extracted.
[12] Intuitively, constantly outputting one and the same index for the empty language is a viable strategy as long as no numerical data has been presented.
[13] Intuitively, changing a learner on inputs that do not present data from a language learned does not harm learnability.

Theorem 3.6. *Let I_0 and I_1 be computably $\mathcal{P}_{\text{c1-1}}$-invariant learning criteria. Suppose I_1 is data normal as witnessed by f_1 and d_1. Let p_0 be such that $W_{p_0} = \emptyset$ and h_1 be such that*

$$\forall x : h_1(x) = \begin{cases} p_0, & \text{if } d_1(x) = \#; \\ \varphi_{d_1(x)}(x), & \text{otherwise.} \end{cases} \tag{12}$$

Further, let $\mathcal{L}_1 = I_1(h_1)$. Then we have

$$[I_1] \setminus [I_0] \neq \emptyset \Leftrightarrow \mathcal{L}_1 \notin I_0.$$

In other words, if I_1 allows for learning a set of languages which is not I_0-learnable, then $\mathcal{L}_1$ is an example such class.

Proof. The implication "$\Leftarrow$" is obvious. Regarding "$\Rightarrow$", let $\mathcal{L}$ be I_1-learnable as witnessed by h and suppose $\mathcal{L}$ is not I_0-learnable. Let Θ be as given by I_1 being computably $\mathcal{P}_{\text{c1-1}}$-invariant. Let $\hat{\Theta}$ be as given by I_1 being data normal. By padded **ORT**, there is a strictly monotone increasing $e \in \mathcal{R}$ such that

$$\forall x, y : \varphi_{e(x)}(y) = (\hat{\Theta} \circ \Theta)(h, e)(y). \tag{13}$$

As $e \in \mathcal{P}_{\text{c1-1}}$ and I_0 is computably $\mathcal{P}_{\text{c1-1}}$-invariant, we have, from Remark 3.2 with I_0 in the place of I, $e(\mathcal{L}) \notin [I_0]$. It now suffices to show $e(\mathcal{L}) \subseteq \mathcal{L}_1$, as this would imply $\mathcal{L}_1 \notin [I_0]$ as desired.

Suppose $I_1 = (\mathcal{C}_1, \beta_1, \delta_1)$. Let $L \in e(\mathcal{L})$ and $T \in \mathbf{Txt}(L)$. We show, by induction on i,

$$\forall i : \beta_1(h_1, T)(i) = \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)(i).$$

Let $h' = \Theta(h, e)$. For all i with $\text{content}(T[i]) = \emptyset$, we have

$$\beta_1(h_1, T)[i] \underset{(8)}{=} h_1(f_1(T[i], \beta_1(h_1, T)[i])) \underset{(12)}{=} p_0 \underset{(9)\,\&\,(10)}{=} \hat{\Theta}(h')(f_1(T[i], \beta_1(\hat{\Theta}(h'), T)[i])) \underset{(8)}{=} \beta_1(\hat{\Theta}(h'), T)(i).$$

Let $i \in \mathbb{N}$, suppose $\text{content}(T[i]) \neq \emptyset$ and (inductively) $\beta_1(h_1, T)[i] = \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)[i]$. Let

$$x = f_1(T[i], \beta_1(h_1, T)[i]) \underset{\text{IH}}{=} f_1(T[i], \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)[i]). \tag{14}$$

Note that $d_1(x) \underset{(9)}{\in} \text{content}(T[i]) \subseteq L \subseteq \text{range}(e)$. We have

$$\beta_1(h_1, T)(i) \underset{(8)\,\&\,(14)}{=} h_1(x) \underset{(12)}{=} \varphi_{d_1(x)}(x) \underset{(13)}{=} (\hat{\Theta} \circ \Theta)(h, e)(x) \underset{(8)\,\&\,(14)}{=} \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)(i).$$

This concludes the induction. Thus, h_1 on any text for a language from $e(\mathcal{L})$ makes the same conjectures as $(\hat{\Theta} \circ \Theta)(h, e)$ on T. By (6) and (7), $(\hat{\Theta} \circ \Theta)(h, e)$ I_1-learns $e(\mathcal{L})$; thus, using (d) of I_1 being data normal, $e(\mathcal{L}) \subseteq I_1(h_1) = \mathcal{L}_1$. □

4. Helping remove U-shapes

In this section we provide, in Theorem 4.4, a general technique for helping with the *removal* of U-shapes from a learner, *preserving what is learned*. When applicable, this shows U-shapes *un*necessary. The main result of this section is Theorem 4.4, which characterizes strongly non-U-shaped learning in terms of sink-stabilizing learning; this theorem is applied in Section 5.

Lemma 4.1. *Let $h \in \mathcal{P}$. Then there is an infinite set $L \in \mathcal{E}$ such that h does not* **TxtGEx***-learn any $L' \supseteq L$ such that $L' \setminus L$ is finite.*

Proof. [14] Trivial if $\mathbb{N} \notin \mathbf{TxtGEx}(h)$. Otherwise, let σ be a locking sequence for h on $\mathbb{N}$, $D = \text{content}(\sigma)$. Then, obviously, h does not learn any L such that $D \subseteq L \subset \mathbb{N}$. In this case, then, let L be an infinite set ce set with infinite complement such that $D \subset L \subset \mathbb{N}$. Clearly this L suffices. □

For each $h \in \mathcal{P}$, let L_h denote a set L corresponding to h and as shown existent in Lemma 4.1.

Let $h \in \mathcal{P}$, $\mathcal{L} = \mathbf{TxtGEx}(h)$ and let Q be a ce predicate on two arguments. Then, using padded s-m-n, there is a strictly monotone increasing function $p_{h,Q} \in \mathcal{R}$ with, for all e, x,

$$W_{p_{h,Q}(e,x)} = \{y \in L_h \mid Q(e, x)\} \cup \{y \in W_e \mid \text{not } (Q(e, x) \text{ in } \leq y \text{ steps})\}.$$

[14] This version of the proof is due to Frank Stephan [40].

Lemma 4.2. *Let $h \in \mathcal{P}$, $\mathcal{L} = \mathbf{TxtGEx}(h)$ and Q be a $\mathtt{ce}$ set. For all e, x, we have*

$$W_{p_{h,Q}(e,x)} \in \mathcal{L} \Rightarrow \neg Q(e,x) \tag{15}$$

$$\Rightarrow W_{p_{h,Q}(e,x)} = W_e. \tag{16}$$

Proof. Immediate from the choice of L_h. □

Let $\beta \in \{\mathbf{G}, \mathbf{Ri}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}$. We have that

$$\{\langle e_0, e_1, e_2\rangle \mid e_0 \text{ is } \textit{not} \text{ a } \beta\text{-sink of } \varphi_{e_1} \text{ on } W_{e_2}\}$$

is $\mathtt{ce}$.

Remember that $\mathcal{R}_{\text{c1-1}} \subseteq \mathcal{R}$ denotes the set of all 1-1 *total* computable functions with computable domain and range.

Definition 4.3. A sequence generating operator β is called *1-1 left-modifiable* iff

$$\forall r \in \mathcal{R}_{\text{c1-1}}\ \exists s_l, s_r \in \mathcal{R}\ \forall h \in \mathcal{P}, T \in \mathbf{Txt} : \beta(s_l \circ h \circ s_r, T) = r \circ \beta(h, T).$$

Note that **G**, **Ri**, **Sd**, **ItCtr** and **It** are 1-1 left-modifiable: for any given $\beta \in \{\mathbf{G}, \mathbf{Ri}, \mathbf{Sd}\}$ and given $r \in \mathcal{R}_{\text{c1-1}}$ we can choose $s_l = r$ and s_r the identity; for the two variants of iterative learning we similarly use $s_l = r$ but now define employ s_r to decode the previous conjecture.

We now present the main theorem of this section, characterizing strongly non-U-shaped learning.

Theorem 4.4. *Let $\beta \preceq \mathbf{G}$ be 1-1 left-modifiable. Let $\mathcal{C} \in \{\mathcal{P}, \mathcal{R}\}$ and let $\mathcal{L} \subseteq \mathcal{R}$. Then the following are equivalent.*

(i) *$\mathcal{L}$ is $\mathcal{C}\mathbf{Txt}\beta\mathbf{SNUEx}$-learnable.*
(ii) *$\mathcal{L}$ is sink-stabilizingly $\mathcal{C}\mathbf{Txt}\beta\mathbf{Ex}$-learnable.*

Proof. "(i) ⇒ (ii)": Let $h_0 \in \mathcal{C}$ be a $\mathbf{Txt}\beta\mathbf{SNUEx}$-learner for $\mathcal{L}$. Let $L \in \mathcal{L}$ and let T be a text for L. Let k be such that h_0 has converged on T after $T[k]$ to some e. Then

$$W_e = L. \tag{17}$$

To show that e is a β-sink of h_0 on L: Let $T \in \mathbf{Txt}(L)$ and i_0 be given such that $\beta(h_0, T)(i_0) = e$. Let $i \geq i_0$. Then, as h_0 is strongly non-U-shaped on L and from (17), $\beta(h_0, T)(i) = e$.

Regarding "(ii) ⇒ (i)": Let $h_0 \in \mathcal{C}$, be a sink-stabilizing $\mathbf{Txt}\beta\mathbf{Ex}$-learner for $\mathcal{L}$. Let Q be a $\mathtt{ce}$ predicate such that

$$\forall e : Q(e) \Leftrightarrow e \text{ is } \textit{not} \text{ a } \beta\text{-sink of } h_0 \text{ on } W_e. \tag{18}$$

Recall the definition of starred learners from Section 2.1, which uses $\beta \preceq \mathbf{G}$. Let $p = p_{h_0^*,Q}$. Let β's left-modifiability with respect to p be witnessed by s_l and $s_r \in \mathcal{R}$. Let $h \in \mathcal{R}$ be such that

$$h = s_l \circ h_0 \circ s_r.$$

Note that, if $h_0 \in \mathcal{R}$, then $h \in \mathcal{R}$.

Claim 1. *h is strongly non-U-shaped.*

Proof of Claim 1. Let $L \in \mathcal{L}$, $e \in \mathbb{N}$ and σ such that $\text{content}(\sigma) \subseteq L$. Suppose $h^*(\sigma) = p(e)$ and

$$W_{p(e)} = L \in \mathcal{L}.$$

From (15) we get $\neg Q(e)$. Hence, from the definition of Q in (18), for all texts T for L extending σ, we have that h_0 has syntactically converged after seeing σ. Thus, h has syntactically converged after seeing σ (and, thus, does not exhibit a U-shape). □ (FOR CLAIM 1)

Claim 2. *h $\mathbf{Txt}\beta\mathbf{Ex}$-learns $\mathcal{L}$.*

Proof of Claim 2. Let $L \in \mathcal{L}$ and let T be a text for L. As h_0 is sink-stabilizing, there is k minimal such that, with $e = h_0^*(T[k])$,

$$e \text{ is a sink of } h \text{ on } L. \tag{19}$$

Thus, h_0 converges on T to e; therefore,

$$W_e = L. \tag{20}$$

Furthermore, h on T converges to $p(e)$ and, by (19), $\neg Q(e)$; hence,

$$W_{p(e)} \underset{(16)}{=} W_e \underset{(20)}{=} L. \quad \square \text{ (for Claim 2)} \quad \square$$

5. Applications of the techniques

In this section we essentially apply general techniques presented in the two just previous sections to prove a number of results pertaining to the necessity of U-shapes in learning.

The following lemma will be useful for later proofs. Parts (i) and (iii) have been shown in [26].

Lemma 5.1. *Let $h \in \mathcal{P}$ and let $\mathcal{C} \in \{\mathcal{P}, \mathcal{R}\}$. Then*

(i) *Any $\mathcal{C}$***TxtGEx***-learnable set can be so learned stabilizingly;*
(ii) *Any* **TxtSdEx***-learner is stabilizing;*
(iii) *Any $\mathcal{C}$***TxtRiEx***-learnable set can be so learned stabilizingly.*

Proof. Claims (i) and (iii) are proven in [26].

Regarding "$\subseteq$" in (ii): Let $h \in \mathcal{P}$. Let $L \in \mathbf{TxtSdEx}(h)$ be a language and suppose σ is a locking sequence of h^* (as defined in Equation (2)) on L. Let T be a text for L. Let c_0 be such that $\text{content}(\sigma) \subseteq \text{content}(T[c_0])$. We show that $T[c_0]$ is a locking sequence of h on L, as desired. In particular, we show, for all $\sigma' \supseteq T[c_0]$ such that $\text{content}(\sigma') \subseteq L$, $h^*(\sigma') = h^*(\sigma)$. Let $\sigma' \supseteq T[c_0]$ such that $\text{content}(\sigma') \subseteq L$.

Let $\sigma'' \supseteq \sigma$ be such that $\text{content}(\sigma'') = \text{content}(\sigma') \subseteq L$. As σ is stabilizing sequence of h^* on L, $h^*(\sigma'') = h^*(\sigma)$. Therefore, we have

$$h^*(\sigma') = h(\text{content}(\sigma')) = h^*(\sigma'') = h^*(\sigma).$$

Thus, h is stabilizing on $\mathcal{L}$. $\square$

Note that [20] implies that $[\mathcal{R}\mathbf{TxtSdEx}] \subset [\mathbf{TxtSdEx}]$. This separation can also be shown using Theorem 3.6. The next theorem shows that, in both cases, we can get strong non-U-shapeness of the learner.

Theorem 5.2. *We have that set-driven learning allows for strongly non-U-shaped learning, i.e.,*

(i) $[\mathbf{TxtSdEx}] = [\mathbf{TxtSdSNUEx}]$ *and*
(ii) $[\mathcal{R}\mathbf{TxtSdEx}] = [\mathcal{R}\mathbf{TxtSdSNUEx}]$.

Proof. We will mostly prove both assertions at once. The inclusion "$\supseteq$" is trivial.

Regarding "$\subseteq$": Let $h_0 \in \mathcal{P}$, let $\mathcal{L} = \mathbf{TxtSdEx}(h_0)$. For all finite sets D, we let $M(D)$ be the set of all $D' \subseteq D$ such that

$$\forall D'' : D' \subseteq D'' \subseteq D \Rightarrow h_0(D'') = h_0(D').$$

Let $h_1 \in \mathcal{P}$ be such that

$$\forall D : h_1(D) = \begin{cases} \uparrow, & \text{if } \exists D' \subseteq D : h_0(D')\uparrow; \\ \text{pad}(h_0(D), \min(M(D))), & \text{otherwise.} \end{cases} \tag{21}$$

To help with Assertion (ii) of the theorem, note that

$$h_0 \in \mathcal{R} \Rightarrow h_1 \in \mathcal{R}. \tag{22}$$

Claim 1. *h_1 is sink-stabilizing on $\mathcal{L}$.*

Proof of Claim 1. Let $L \in \mathcal{L}$, let $T \in \mathbf{Txt}(L)$. Using Lemma 5.1, h_0 stabilizes on $\mathcal{L}$; thus, let $D_0 \subseteq L$ be the $\leq$-minimum such that

$$(\forall D \mid D_0 \subseteq D \subseteq L) h_0(D) = h_0(D_0). \tag{23}$$

Then we have

$$\forall^{\infty} n : h_1(\text{content}(T[n])) = \text{pad}(h_0(D_0), D_0). \tag{24}$$

Furthermore, for all D such that $h_1(D) = \text{pad}(h_0(D_0), D_0)$ we have $D \supseteq D_0$. □ (FOR CLAIM 1)

The theorem now follows from Theorem 4.4. □

With a similar proof as in Theorem 5.2, we can also show that rearrangement-independent learning allows for strongly non-U-shaped learning.

Theorem 5.3. *We have that rearrangement-independent learners can be assumed total and strongly non-U-shaped, i.e.,*

$$[\mathcal{R}\textbf{TxtRiSNUEx}] = [\textbf{TxtRiEx}].$$

Proof. The inclusion "$\supseteq$" is trivial.

"$\subseteq$": Let $h_0 \in \mathcal{P}$, let $\mathcal{L} = \textbf{TxtRiEx}(h_0)$. From [26] we know that, without loss of generality, $h_0 \in \mathcal{R}$. Thanks to Lemma 5.1, we can suppose, without loss of generality, h_0 is stabilizing on $\mathcal{L}$. For all D', c', D, c, we write $\langle D', c'\rangle \to \langle D, c\rangle$ iff there is a text T such that $\text{content}(T[c']) = D'$, $\text{content}(T[c]) = D$ and $c' \leq c$ (i.e., it is possible that at some point $\langle D', c'\rangle$ is the input to a learner, and at some later point $\langle D, c\rangle$ is the input).

Let R be a computable predicate such that, for all D', c', D, c, $R(D', c', D, c)$ iff $\langle D', c'\rangle \to \langle D, c\rangle$ and

$$\forall \langle D'', c''\rangle : (\langle D', c'\rangle \to \langle D'', c''\rangle \to \langle D, c\rangle) \Rightarrow h_0(D'', c'') = h_0(D, c).$$

We define computable predicates $f, h_1 \in \mathcal{P}$ such that

$$\forall D, c : f(D, c) = \min\{\langle D', c'\rangle \mid R(D', c', D, c)\};^{15}$$

$$\forall D, c : h_1(D, c) = \text{pad}(h_0(f(D, c)), f(D, c)).$$

Note that $h_1 \in \mathcal{R}$. Further note that, for all x, $\varphi_{h_0(x)} = \varphi_{h_1(x)}$. Of course we have, as h_0 is stabilizing on $\mathcal{L}$, for all $L \in \mathcal{L}$ and all texts T for L that there is a c with

$$(\forall D', c' | \langle \text{content}(T[c]), c\rangle \to \langle D', c'\rangle \wedge D' \subseteq L)\, h_0(\text{content}(T[c]), c) = h_0(D', c').$$

Hence, h_1 is stabilizing on $\mathcal{L}$, and, thus,

$$\mathcal{L} \subseteq \textbf{TxtRiEx}(h_1).$$

Claim 1. *h_1 is sink-stabilizing.*

Proof of Claim 1. Let $L \in \mathcal{L}$ and let T be a text for L. Let $D \subseteq L$ and c be such that h_1 stabilizes on $\rho \subseteq T$ to $\text{pad}(h_0(D, c), \langle D, c\rangle)$, i.e.,

$$(\forall \langle D', c'\rangle \mid \langle \text{content}(\rho), \text{len}(\rho)\rangle \to \langle D', c'\rangle) h_1(D', c') = \text{pad}(h_0(D, c), \langle D, c\rangle). \tag{25}$$

It remains to show that $\text{pad}(h_0(D, c), \langle D, c\rangle)$ is a sink for h_1 on L, i.e., that for all ρ' with $\text{content}(\rho') \subseteq L$ we have $h_1^*(\rho \diamond \rho') = \text{pad}(h_0(D, c), \langle D, c\rangle)$. Let ρ' be such that $\text{content}(\rho') \subseteq L$. We have $\langle D, c\rangle \to \langle \text{content}(\rho), \text{len}(\rho)\rangle$. Therefore, $\langle D, c\rangle \to \langle \text{content}(\rho \diamond \rho'), \text{len}(\rho \diamond \rho')\rangle$, and thus, using (25), $h_1^*(\rho \diamond \rho') = \text{pad}(h_0(D, c), \langle D, c\rangle)$. □ (FOR CLAIM 1)

An application of Theorem 4.4 finishes the proof. □

As $[\textbf{TxtGEx}] = [\textbf{TxtRiEx}]$ [37,25,26,28], we immediately get the following corollary, reproving a result from [3] and one from [21].

Corollary 5.4. *We have that full-information learning allows for strongly non-U-shaped learning, and thus also for non-U-shaped learning, i.e.,*

$$[\textbf{TxtGNUEx}] \underset{[3]}{=} [\textbf{TxtGEx}] \underset{[21]}{=} [\textbf{TxtGSNUEx}].$$

The following definition and the succeeding lemma are from [19]. These will be used for the proof of the final theorem for this paper, Theorem 5.7.

[15] Note that, for all D and c such that $\text{card}(D) \geq c$, $R(D, c, D, c)$.

Definition 5.5. For all $M \in \mathcal{P}$, M is called *canny* iff, for all σ, (i) through (iii) below (recall from Section 2.1 that M^* is the **G**-learner equivalent to the iterative learner M).

(i) $M^*(\sigma)\downarrow \Rightarrow M^*(\sigma) \in \mathbb{N}$, i.e., M^* never outputs ?.
(ii) $M^*(\sigma \diamond \#) = M^*(\sigma)$.
(iii) For all $x \in \mathbb{N}$, if $M^*(\sigma \diamond x) \neq M^*(\sigma)$, then, for all $\tau \supseteq \sigma \diamond x$, $M^*(\tau \diamond x) = M^*(\tau)$.

Lemma 5.6. *[Theorem 1 in [19]] For all $M' \in \mathcal{P}$, there is M such that* **TxtItEx**$(M') \subseteq$ **TxtItEx**(M) *and M is canny.*

From the proof of [19, Theorem 1], it is easy to see that, if M' is *strongly* non-U-shaped, then so is M. Note that the construction in that proof does not, in general, preserve non-U-shapedness.

From [19, Theorem 2] we have [**TxtItEx**] = [**TxtItNUEx**]. Contrasting this result, we have the following theorem.

Theorem 5.7. *We have that there are non-U-shapedly iteratively learnable collections of languages which are not strongly non-shapedly learnable by an iterative learner, i.e.,*

[**TxtItSNUEx**] $\subset$ [**TxtItNUEx**].

Our proof makes use of padded **ORT** and, as noted above, for convenience, a self-learning class simpler to work with than that from Theorem 3.6.

Proof. It is easy to see that, in our present setting, the initial conjecture of any given iterative learner does not matter. Hence, for the following learner M, we will suppose the initial conjecture to be $\mathrm{pad}(0, \emptyset)$. Let $M \in \mathcal{P}$ be such that

$$\forall e, D, x : M(\mathrm{pad}(e, D), x) = \begin{cases} \mathrm{pad}(e, D), & \text{if } x \in D \cup \{\#\}; \\ \uparrow, & \text{else if } \varphi_x(e)\uparrow; \\ \mathrm{pad}(\varphi_x(e), D \cup \{x\}), & \text{else if } \varphi_x(e) \neq e; \\ \mathrm{pad}(e, D), & \text{otherwise.} \end{cases}$$

Obviously, M is canny. Let $\mathcal{L} =$ **TxtItEx**(M). From [19, Theorem 2] we have $\mathcal{L}$ is **TxtItNUEx**-learnable.

We proceed by showing that no canny **TxtItEx**-learner for $\mathcal{L}$ can be strongly non-U-shaped. By the remark after Lemma 5.6, any strongly non-U-shaped **TxtItEx**-learner can be assumed canny; hence, there is no **TxtItSNUEx**-learner for $\mathcal{L}$ if there is no canny **TxtItSNUEx**-learner for $\mathcal{L}$. Let $h \in \mathcal{P}$ be canny such that $\mathcal{L} \subseteq$ **TxtItEx**(h).

Let p be a program for h^* (where h^* is again a starred learner as introduced in Section 2.1). We will make use of a *Blum complexity measure* [4] Φ associated with φ; intuitively, for all programs p and inputs x, $\varphi_p(x)\uparrow$ iff $\Phi_p(x)\uparrow$, and the predicate $(p, x, t) \mapsto \Phi_p(x) \leq t$ is decidable (one can think of $\Phi_p(x)$ as the number of steps until program p holds on input x). By the padded **ORT**, there are $a, b, e_1, e_2 \in \mathcal{R}$ strictly monotone increasing with pairwise disjoint ranges and $e_0 \in \mathbb{N}$ such that, abbreviating, for all i, k,

$$\begin{aligned} P_k(i) &\Leftrightarrow h^*(a[k] \diamond b(k) \diamond a(i))\downarrow = h^*(a[k] \diamond b(k))\downarrow, \\ \overline{P}_k(i) &\Leftrightarrow h^*(a[k] \diamond b(k) \diamond a(i))\downarrow \neq h^*(a[k] \diamond b(k))\downarrow, \\ E_k &= \left\{ a(i) \;\middle|\; \begin{array}{l} \exists j[(\forall j' < j : P_k(j')) \wedge \Phi_p(a[k] \diamond b(k) \diamond a(j)) \geq i] \\ \vee\, \forall j < i : P_k(j) \end{array} \right\}, \end{aligned}$$

we have, for all e, i, k,

$$\varphi_{a(i)}(e) = \begin{cases} e_0, & \text{if } e \in \{?, e_0\}; \\ e_2(k), & \text{else if } e = e_1(k) \text{ and } i \geq k; \\ e, & \text{otherwise;} \end{cases}$$

$$\varphi_{b(k)}(e) = \begin{cases} e_1(k), & \text{if } e \in \{?, e_0\}; \\ e, & \text{otherwise;} \end{cases}$$

$$\begin{aligned} W_{e_0} &= \mathrm{content}(a); \\ W_{e_1(k)} &= \mathrm{content}(a[k]) \cup \{b(k)\} \cup \begin{cases} E_k, & \text{if } \exists i : (\overline{P}_k(i) \wedge \forall j < i : P_k(j)); \\ \emptyset, & \text{otherwise;} \end{cases} \\ W_{e_2(k)} &= \mathrm{content}(a[k]) \cup \{b(k)\} \cup E_k. \end{aligned}$$

It is easy to see that $W_{e_0} \in \mathcal{L}$ as witnessed by M. Hence, h on the text a for W_{e_0} converges to a correct program number for W_{e_0}. Let k be such that

$$\forall k' \geq k : h^*(a[k']) = h^*(a[k])$$

and

$$W_{h^*(a[k])} = W_{e_0}.$$

We now focus on the sets $W_{e_1(k)}$ and $W_{e_2(k)}$.

Claim 1. *There is i such that $\overline{P}_k(i)$ and $\forall j < i : P_k(j)$.*

Proof of Claim 1. Suppose, by way of contradiction, otherwise, i.e., suppose

$$\neg[\exists i : (\overline{P}_k(i) \wedge \forall j < i : P_k(j))]. \tag{26}$$

This is equivalent to

$$\forall i : [P_k(i) \vee h^*(a[k] \diamond b(k) \diamond a(i))\uparrow \vee \exists j < i : \neg P_k(j)]. \tag{27}$$

Which is equivalent to

$$[\forall i : P_k(i)] \vee [\exists i_0 : (h^*(a[k] \diamond b(k) \diamond a(i_0))\uparrow \wedge \forall j < i_0 : P_k(j))]. \tag{28}$$

Note that (28) is implied by (27) by choosing the i_0 in the second disjunct of (28) to be the minimum i making the first disjunct of (27) false, if any.

Subclaim: $E_k = \text{content}(a)$.

We proceed by considering possible cases of (28).
Suppose first that $\forall i : P_k(i)$. Then, by the second disjunct in the definition of E_k, $E_k = \text{content}(a)$.
Suppose second that there is i_0 such that $h^*(a[k] \diamond b(k) \diamond a(i_0))\uparrow$ and, for all $j < i_0$, $P_k(j)$. Then, by the first disjunct in the definition of E_k, we get the subclaim. □ (FOR SUBCLAIM)

Using (26), we have that

$$W_{e_1(k)} = \text{content}(a[k]) \cup \{b(k)\}.$$

Further, by the Subclaim,

$$W_{e_2(k)} = \text{content}(a[k]) \cup \{b(k)\} \cup \text{content}(a).$$

It is easy to see that $W_{e_1(k)}$ and $W_{e_2(k)}$ are elements of $\mathcal{L}$. Note that $a[k] \diamond b(k) \diamond \lambda i.\#$ is a text for $W_{e_1(k)}$, and, as h is canny, h converges on this text to $h^*(a[k] \diamond b(k))$. Also note that $a[k] \diamond b(k) \diamond a$ is a text for $W_{e_2(k)}$, and, using (26) and the fact that h is an iterative learner, h on this text converges to $h^*(a[k] \diamond b(k))$. This is a contradiction to $W_{e_2(k)} \neq W_{e_1(k)}$, as h supposedly identifies both. □ (FOR CLAIM 1)

Let i_0 be the minimum i_0 as shown existent by Claim 1. Then E_k is finite, and, in particular, there is k' such that $E_k = \text{content}(a[k'])$. Let $k_1 = \max(k, k')$. Letting $L = \text{content}(a[k_1]) \cup \{b(k)\}$, we have that

$$W_{e_1(k)} = L = W_{e_2(k)}$$

is a finite set. It is easy to see that $L \in \mathcal{L}$. Note that

$$a(i_0) \in L. \tag{29}$$

As h is canny, we get, as $a[k_1] \diamond b(k) \diamond \lambda i.\#$ is a text for L,

$$W_{h^*(a[k_1] \diamond b(k))} = L.$$

However, by choice of i_0,

$$h^*(a[k_1] \diamond b(k)) \neq h^*(a[k_1] \diamond b(k) \diamond a(i_0)).$$

Using (29), we see that h is *not* strongly non-U-shaped. □

6. Conclusion and open problems

Summing up, the following table indicates whether, for a given learning criterion (first column) semantic U-shapes (second column) or syntactic U-shapes (third column) can be avoided; in particular, "yes" in a column of the row of a learning criterion I means that every I-learnable set is I-learnable by a (strongly) non-U-shaped learner. Some of the justificatory remarks omit citing [**TxtGEx**] = [**TxtRiEx**] shown in [37,26].

	NU	**SNU**
TxtGEx	yes, [3], Corollary 5.4	yes, [21], Corollary 5.4
$\mathcal{R}$**TxtGEx**	yes, Theorem 5.3	yes, [3], Theorem 5.3
TxtRiEx	yes, Theorem 5.3	yes, Theorem 5.3
$\mathcal{R}$**TxtRiEx**	yes, Theorem 5.3	yes, Theorem 5.3
TxtSdEx	yes, Theorem 5.2	yes, Theorem 5.2
$\mathcal{R}$**TxtSdEx**	yes, Theorem 5.2	yes, Theorem 5.2
TxtItEx	yes, [19]	no, Theorem 5.7
$\mathcal{R}$**TxtItEx**	*open*	no, [17]

We think that one can obtain [$\mathcal{R}$**TxtItNUEx**] = [$\mathcal{R}$**TxtItEx**] as a corollary to the proof in [19] of [**TxtItNUEx**] = [**TxtItEx**]. Trivially, we have

$$[\textbf{TxtItCtrSNUEx}] \subseteq [\textbf{TxtItCtrNUEx}] \subseteq [\textbf{TxtItCtrEx}];$$
$$[\mathcal{R}\textbf{TxtItCtrSNUEx}] \subseteq [\mathcal{R}\textbf{TxtItCtrNUEx}] \subseteq [\mathcal{R}\textbf{TxtItCtrEx}].$$

The other directions of inclusions are *open*.

As we can see from the overview, U-shaped learning is unnecessary in many different learning settings. Only very restricted settings in which learners make use of the hypothesis as a way to store information (as in the case of iterative learning), can sometimes require U-shapes.

Acknowledgments

We would like to thank the anonymous referees both of the conference version and the journal version of this paper for their comments and suggestions. Further, we want to thank Samuel E. Moelius III and Thomas Zeugmann for discussions on various theorems and their proofs. Major work on this paper was conducted while the second author was at the Department of Computer and Information Sciences, University of Delaware, USA.

References

[1] J.M. Barzdin, Two Theorems on the Limiting Synthesis of Functions, Theory of Algorithms and Programs I, vol. 210, Latvian State University, Riga, 1974, pp. 82–88 (in Russian).
[2] Lenore Blum, Manuel Blum, Toward a mathematical theory of inductive inference, Inf. Control 28 (2) (1975) 125–155.
[3] Ganesh Baliga, John Case, Wolfgang Merkle, Frank Stephan, Rolf Wiehagen, When unlearning helps, Inf. Comput. 206 (5) (2008) 694–709.
[4] Manuel Blum, A machine-independent theory of the complexity of recursive functions, J. ACM 14 (2) (1967) 322–336.
[5] T. Bower, Concepts of development, in: Proceedings of the 21st International Congress of Psychology, Presses Universitaires de France, Paris, 1978.
[6] M. Bower, Starting to talk worse: clues to language development from children's late speech errors, in: S. Strauss, R. Stavy (Eds.), U-Shaped Behavioral Growth, in: Developmental Psychology Series, Academic Press, NY, 1982.
[7] S. Carey, Face perception: anomalies of development, in: S. Strauss, R. Stavy (Eds.), U-Shaped Behavioral Growth, in: Developmental Psychology Series, Academic Press, NY, 1982.
[8] John Case, Periodicity in generations of automata, Math. Syst. Theory 8 (1) (1974) 15–32.
[9] John Case, Infinitary self-reference in learning theory, J. Exp. Theor. Artif. Intell. 6 (1) (1994) 3–16.
[10] John Case, The power of vacillation in language learning, SIAM J. Comput. 28 (6) (1999) 1941–1969.
[11] Lorenzo Carlucci, John Case, Sanjay Jain, Frank Stephan, Non-u-shaped vacillatory and team learning, J. Comput. Syst. Sci. 74 (4) (2008) 409–430.
[12] John Case, Sanjay Jain, Steffen Lange, Thomas Zeugmann, Incremental concept learning for bounded data mining, Inf. Comput. 152 (1) (1999) 74–110.
[13] John Case, Timo Kötzing, Dynamically delayed postdictive completeness and consistency in learning, in: Yoav Freund, László Györfi, György Turán, Thomas Zeugmann (Eds.), Algorithmic Learning Theory, 19th International Conference, ALT 2008, Proceedings, Budapest, Hungary, October 13–16, 2008, in: Lecture Notes in Computer Science, vol. 5254, Springer, 2008, pp. 389–403.
[14] John Case, Timo Kötzing, Strongly non-u-shaped learning results by general techniques, in: Adam Tauman Kalai, Mehryar Mohri (Eds.), COLT 2010 – The 23rd Conference on Learning Theory, Haifa, Israel, June 27–29, 2010, Omnipress, 2010, pp. 181–193.
[15] John Case, Timo Kötzing, Computability-theoretic learning complexity, Philos. Trans. R. Soc. A, Math. Phys. Eng. Sci. 370 (2012) 3570–3596.
[16] John Case, Timo Kötzing, Memory-limited non-u-shaped learning with solved open problems, Theor. Comput. Sci. 473 (2013) 100–123.
[17] John Case, Timo Kötzing, Topological separations in inductive inference, Theor. Comput. Sci. 620 (2016) 33–45.
[18] John Case, Christopher Lynes, Machine inductive inference and language identification, in: Mogens Nielsen, Erik Meineche Schmidt (Eds.), Automata, Languages and Programming, 9th Colloquium, Proceedings, Aarhus, Denmark, July 12–16, 1982, in: Lecture Notes in Computer Science, vol. 140, Springer, 1982, pp. 107–115.
[19] John Case, Samuel E. Moelius, U-shaped, iterative, and iterative-with-counter learning, Mach. Learn. 72 (1–2) (2008) 63–88.

[20] John Case, Samuel E. Moelius, Parallelism increases iterative learning power, Theor. Comput. Sci. 410 (19) (2009) 1863–1875.
[21] John Case, Samuel E. Moelius, Optimal language learning from positive data, Inf. Comput. 209 (10) (2011) 1293–1311.
[22] John Case, Carl Smith, Comparison of identification criteria for machine inductive inference, Theor. Comput. Sci. 25 (1983) 193–220.
[23] Jerome A. Feldman, Some decidability results on grammatical inference and complexity, Inf. Control 20 (3) (1972) 244–262.
[24] Mark A. Fulk, Sanjay Jain, Daniel N. Osherson, Open problems in "systems that learn", J. Comput. Syst. Sci. 49 (3) (1994) 589–604.
[25] M. Fulk, A study of inductive inference machines, PhD thesis, SUNY at Buffalo, 1985.
[26] Mark A. Fulk, Prudence and other conditions on formal language learning, Inf. Comput. 85 (1) (1990) 1–11.
[27] E. Mark Gold, Language identification in the limit, Inf. Control 10 (5) (1967) 447–474.
[28] S. Jain, D. Osherson, J. Royer, A. Sharma, Systems that Learn: An Introduction to Learning Theory, second edition, MIT Press, Cambridge, Mass., 1999.
[29] 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.
[30] Timo Kötzing, Iterative learning from positive data and counters, Theor. Comput. Sci. 519 (2014) 155–169.
[31] G. Marcus, S. Pinker, M. Ullman, M. Hollander, T.J. Rosen, F. Xu, Overregularization in Language Acquisition, Monographs of the Society for Research in Child Development, vol. 57(4), University of Chicago Press, 1992, includes commentary by H. Clahsen.
[32] P. Odifreddi, Classical Recursion Theory, vol. II, Elsevier, Amsterdam, 1999.
[33] Daniel N. Osherson, Scott Weinstein, Criteria of language learning, Inf. Control 52 (2) (1982) 123–138.
[34] K. Plunkett, V. Marchman, U-shaped learning and frequency effects in a multi-layered perceptron: implications for child language acquisition, Cognition 38 (1) (1991) 43–102.
[35] J. Royer, J. Case, Subrecursive Programming Systems: Complexity and Succinctness, research monograph in Progress in Theoretical Computer Science, Birkhäuser, Boston, 1994.
[36] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw Hill, New York, 1967, reprinted by MIT Press, Cambridge, Massachusetts, 1987.
[37] G. Schäfer-Richter, Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien, Dissertation, RWTH Aachen, 1984.
[38] S. Strauss, R. Stavy (Eds.), U-Shaped Behavioral Growth, Developmental Psychology Series, Academic Press, NY, 1982.
[39] S. Strauss, R. Stavy, N. Orpaz, The child's development of the concept of temperature, Tel-Aviv University, 1977, unpublished manuscript.
[40] F. Stephan, 2009, private communication.
[41] N.A. Taatgen, J.R. Anderson, Why do children learn to say broke? a model of learning the past tense without feedback, Cognition 86 (2) (2002) 123–155.
[42] K. Wexler, P. Culicover, Formal Principles of Language Acquisition, MIT Press, Cambridge, Mass., 1980.
[43] Rolf Wiehagen, A thesis in inductive inference, in: Jürgen Dix, Klaus P. Jantke, Peter H. Schmitt (Eds.), Nonmonotonic and Inductive Logic, 1st International Workshop, Proceedings, Karlsruhe, Germany, December 4–7, 1990, in: Lecture Notes in Computer Science, vol. 543, Springer, 1991, pp. 184–207.