Theoretical Computer Science 519 (2014) 155–169

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Iterative learning from positive data and counters

Timo Kötzing

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

ARTICLE INFO

Keywords:
Inductive inference
Function learning
Iterative learning
Transductive learning

ABSTRACT

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

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

1. Introduction

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

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

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

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

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

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

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

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

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

We define the following counter types[2]:

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

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

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

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

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

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