Limits of Computational Learning

Timo Kötzing

July 22, 2012

Abstract

$3, 5, 7, 11, 13, \ldots$ – what's next? What general rule (apparently) produces this sequence? Maybe the sequence lists all the odd primes, but what if the next datum is 15? Maybe all odd numbers that are not squares? In this course we will study learning (identification) of infinite objects (such as infinite sequences) from finite data (such as initial pieces of the sequence), also known as *Inductive Inference*. What (collections of) sequences can be learned? What does learning, or identification, actually mean? We will discuss and compare several notions of "identification." The main focus lies on exploring the limits of what can be learned algorithmically.

1 Introduction

This course starts with a review of some notions from computability theory (Section 2). After this, we will consider some of the main results from computational learning theory. A standard reference for learning theory is [JORS99].

2 Mathematical Preliminaries

In this section we introduce our mathematical notation and review basic computability theory. A gentle and quick introduction, also covering details of the machine model and basic theorems, is given in [Sho01]. A standard reference for computability in general is [Rog67], which provides much greater depth than [Sho01]. Details regarding Turingmachines and their coding, as well as details for time and space complexity can be found in [RC94].

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

We fix any 1-1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Define π_1 and π_2 to be the functions such that, for all x and y,

$$\pi_1(\langle x, y \rangle) = x; \tag{1}$$

$$\pi_2(\langle x, y \rangle) = y. \tag{2}$$

π_1 and π_2 are, respectively, called the first and second projection function.

The symbols $\subseteq, \subset, \supseteq, \supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets. For sets A, B, we let $A \setminus B = \{a \in A \mid a \notin B\}$, $\overline{A} = \mathbb{N} \setminus A$ and $\mathrm{Pow}(A)$ be the power set of A.

The quantifier $\forall^\infty x$ means "for all but finitely many x", the quantifier $\exists^\infty x$ means "for infinitely many x". For any set A, $\mathrm{card}(A)$ denotes the cardinality of A.

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.

For two partial functions f, g and $a \in \mathbb{N}$, we write $f =^a g$ iff $\mathrm{card}(\{x \in \mathbb{N} \mid f(x) \neq g(x)\}) \leq a$; we write $f =^* g$ iff $\mathrm{card}(\{x \in \mathbb{N} \mid f(x) \neq g(x)\})$ is finite.

For two partial functions f, g, we let $f \circ g = \lambda x . f(g(x))$.

Whenever we consider tuples of natural numbers as input to f, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to (left-associatively) code the tuples into a single natural number.

If 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 $f(x)\downarrow = p$.

For a function f and a natural number n we let $f[n]$ denote the finite sequence $f(0), \ldots, f(n-1)$ (undefined, if one of these is undefined).

2.1 Computabiliy Notions

A partial function $f : \mathbb{N} \to \mathbb{N}$ is *partial computable* iff there is a deterministic, multi-tape Turing machine which, on input x, returns $f(x)$ if $f(x)\downarrow$, and loops infinitely if $f(x)\uparrow$. The set of all (partial) computable functions is denoted with $\mathcal{P}$. The total computable functions are denoted with $\mathcal{R}$.

For a given (code of a) Turing machine program p, we let φ_p denote the function computed by (the Turing machine corresponding to) p.

Thus, for each p, φ_p is an element of $\mathcal{P}$, and all elements of $\mathcal{P}$ can be so represented.

For each p, x, we let $\Phi_p(x)$ denote the number of computation steps the Turing machine (coded by) p takes on input x (if it terminates, undefined otherwise); thus, Φ_p denotes the partial computable *runtime* function of the

TM-program with code number p in the φ-system.

A set of natural numbers is called *decidable* iff its characteristic function is computable.

An *effective operator* is a mapping $\Theta : \mathcal{P} \to \mathcal{P}$ such that there is an $f \in \mathcal{R}$ with, for all e, $\Theta(\varphi_e) = \varphi_{f(e)}$; similarly for Θ of higher arity.

We give a number of important theorems of computability theory; we omit the proofs, which can be found in [Sho01] and, in more detail, in [RC94].

Theorem 2.1 (Characterization of Computable Functions)**.** The class of computable functions is the smallest class $\mathcal{C}$ such that

- the functions $\lambda x.0$ and $\lambda x.x+1$ are in $\mathcal{C}$ and, for all i, k with $i \leq k$, $\lambda x_0, \ldots, x_k.x_i$ are in $\mathcal{C}$;
- for all $f, g_0, \ldots, g_n \in \mathcal{C}$, $\lambda x.f(g_0(x), \ldots, g_n(x))$ is in $\mathcal{C}$;
- $\mathcal{C}$ is closed under "inductive definition;"
- if $f \in \mathcal{C}$ is a predicate, then $\lambda x.\min_y f(x,y)$ is in $\mathcal{C}$.[1]

Theorem 2.2 (Further Closure Properties)**.** The class of computable functions is closed under

- logical connectives;
- arithmetic operations and relations;
- coding of tuples;
- bounded quantification;
- if-then-else;
- finite variants.

Note that the set of *total* computable functions is also closed under the above operations, except the minimization; here it is only closed if an explicit minimization bound is given.

Theorem 2.3 (Halting Problem)**.** The predicate

$$\lambda p, x.\varphi_p(x)\downarrow$$

is *not* a total computable predicate.

Theorem 2.4 (Blum Complexity Measure)**.** The predicate

$$\lambda p, x, t.\Phi_p(x) \leq t$$

is total computable.

Theorem 2.5 (Universality)**.** There is a program u such that, for all p, x,

$$\varphi_u(p,x) = \varphi_p(x).$$

[1] Note that $\min_y f(x,y)$ is defined iff there is a y such that $f(x,y)$ *and*, for all $y' < y$, $f(x,y)\downarrow$.

Theorem 2.6 (Padding Theorem)**.** There is a 1-1 total computable function pad such that, for all p, x,

$$\varphi_{\mathrm{pad}(p,x)} = \varphi_p.$$

Furthermore, pad can be chosen strictly monotone increasing in both arguments.

We fix such a function pad for these lecture notes. We will use it to get a program equivalent to some given program e and different from a given value x.

Theorem 2.7 (S-m-n Theorem, Parameter Theorem)**.** There is a total computable function s such that, for all e, x, y,

$$\varphi_{s(e,x)}(y) = \varphi_e(x,y).$$

Furthermore, s can be chosen to be strictly monotone increasing. In particular, for all computable functions f there is a strictly monotone increasing total computable function s such that, for all x, y,

$$\varphi_{s(x)}(y) = f(x,y).$$

Theorem 2.8 (Kleene's Recursion Theorem, **KRT**)**.** There is a total computable function e such that, for all p, x,

$$\varphi_{e(p)}(x) = \varphi_p(e(p), x).$$

In particular, for all (partial) computable functions f, there is an e such that

$$\varphi_e(x) = f(e,x).$$

Theorem 2.9 (Rogers' Fixpoint Theorem)**.** For all total computable functions f there is an e such that

$$\varphi_{f(e)} = \varphi_e.$$

Theorem 2.10 (Case's Operator Recursion Theorem, ORT, [Cas74])**.** Let Θ be an effective operator. Then there is a total computable function e such that, for all n, x,

$$\varphi_{e(n)}(x) = \Theta(e)(n,x).$$

Furthermore, e can be chosen strictly monotone increasing.

The importance of e in the theorem just above being strictly monotone increasing is given in the following proposition.

Proposition 2.11. Let $f \in \mathcal{R}$ be strictly monotone increasing. Then range(f) is decidable.

The proof of this proposition is left as an exercise.

2.2 Examples

In this section we give examples for the usage of theorems from Section 2.1.

Theorem 2.12. There is a total computable function r such that, for all finite sequences of natural numbers σ, e and x,

$$\varphi_{r(\sigma,e)}(x) = \begin{cases} \sigma(x), & \text{if } x < \text{len}(\sigma); \\ \varphi_e(x), & \text{otherwise.} \end{cases}$$

For two sets A, B of natural numbers we write $A \leq_1 B$ iff there is a 1-1 total computable function f such that, for all e, $e \in A$ iff $f(e) \in B$. Note that this induces a preorder on the powerset of $\mathbb{N}$; intuitively, sets higher up in the order are more complicated to decide algorithmically. For $A, B \subseteq \mathbb{N}$ we write $A \equiv_1 B$ iff $A \leq_1 B$ and $B \leq_1 A$.

We define the following sets of natural numbers.

$$\begin{aligned} A_1 &= \{e \in \mathbb{N} \mid \varphi_e(0)\downarrow\}; \\ A_2 &= \{e \in \mathbb{N} \mid \varphi_e \text{ is total}\}; \\ A_3 &= \{e \in \mathbb{N} \mid \text{dom}(\varphi_e) \text{ is infinite}\}; \\ A_4 &= \{e \in \mathbb{N} \mid \text{dom}(\varphi_e) \neq \emptyset\}; \\ A_5 &= \{e \in \mathbb{N} \mid \text{range}(\varphi_e) \text{ is infinite}\}; \\ A_6 &= \{e \in \mathbb{N} \mid \exists a : \forall^\infty t : \varphi_e(t) = a\}. \end{aligned}$$

Theorem 2.13. We have that

$$A_1 \equiv_1 A_4 \leq_1 A_2 \equiv_1 A_3 \equiv_1 A_5 \leq_1 A_6.$$

Proof. We first prove $A_1 \leq_1 A_4$. By universality, there is a program p such that, for all e, x $\varphi_p(e, x) = \varphi_e(0)$. Let s be as given by the S-m-n Theorem (Theorem 2.7). Let $f = \lambda e.s(p, e)$. Then r is 1-1 and computable. We need to show that, for all e, $e \in A_1$ iff $f(e) \in A_4$.

Let $e \in \mathbb{N}$ be given. We have, for all x,

$$\varphi_{f(e)}(x) = \varphi_{s(p,e)}(x) = \varphi_p(e, x) = \varphi_e(0).$$

Suppose first $e \in A_1$. We get $\varphi_e(0)\downarrow$; hence, $\varphi_{f(e)}(0)\downarrow$, showing $0 \in \text{dom}(\varphi_{f(e)})$, which gives $f(e) \in A_4$ as desired.

Suppose second $f(e) \in A_4$. We get $\exists x : \varphi_{f(e)}(x)\downarrow$; pick any such x. We have $\varphi_{f(e)}(x) = \varphi_e(0)$, which shows $\varphi_e(0)\downarrow$, thus $e \in A_1$ as desired.

We now prove $A_4 \leq_1 A_1$. Using the above closure properties and Theorem 2.4, there is a program p such that, for all e, x

$$\varphi_p(e, x) = \mu t.\exists a \leq t : \Phi_e(a) \leq t.$$

Let s be again as given by the S-m-n Theorem (Theorem 2.7). Let $f = \lambda e.s(p, e)$. Then r is 1-1 and computable. We need to show that, for all e, $e \in A_4$ iff $f(e) \in A_1$.

Let $e \in \mathbb{N}$ be given. We have, for all x,

$$\varphi_{f(e)}(x) = \varphi_{s(p,e)}(x) = \varphi_p(e, x) = \mu t.\exists a \leq t : \Phi_e(a) \leq t.$$

Suppose first $e \in A_4$ and let $x \in \text{dom}(\varphi_e)$. Further, let $t = \Phi_e(x)$. This t is an example t such that $\exists a \leq t : \Phi_e(a) \leq t$, which shows $\varphi_{f(e)}(0)\downarrow$, and, thus, $f(e) \in A_1$ as desired.

Suppose second $f(e) \in A_1$. Thus, $\varphi_{f(e)}(0)\downarrow$. Therefore, there is a t such that $\exists a \leq t : \Phi_e(a) \leq t$; pick any such t and let a be such that $\Phi_e(a) \leq t$. Then we have $a \in \text{dom}(\varphi_e)$, which gives $e \in A_4$ as desired.

Regarding $A_2 \leq_1 A_3$ we use the S-m-n Theorem similarly as above to get a 1-1 $f \in \mathcal{R}$ such that, for all e, x,

$$\varphi_{f(e)}(x) = \mu t.\forall a \leq x : \Phi_e(a) \leq t.$$

Let $e \in \mathbb{N}$ be given. Suppose first $e \in A_2$. We get φ_e is total; hence, for all x, there is a t such that $\forall a \leq x : \Phi_e(a) \leq t$. This gives that $\varphi_{f(e)}$ is total and, thus, has infinite domain. Therefore we have $f(e) \in A_4$ as desired.

Suppose second that $f(e) \in A_3$ and let $x \in \mathbb{N}$. It suffices to show that φ_e is defined on x (as x is chosen arbitrarily, this shows φ_e to be total, i.e., $e \in A_2$). Let $y > x$ such that $y \in \text{dom}(\varphi_{f(e)})$ (this has to exist since $f(e) \in A_3$). Thus, there is a t such that $\forall a \leq y : \Phi_e(a) \leq t$. In particular, for $a = x$, we get $\Phi_e(x) \leq t$, which gives $\varphi_e(x)\downarrow$ as desired.

Regarding $A_1 \leq_1 A_2$ we use f as in the proof for $A_1 \leq_1 A_4$, in particular, f is such that, for all e, x,

$$\varphi_{f(e)}(x) = \varphi_e(0).$$

Let $e \in \mathbb{N}$ be given.

Suppose first $e \in A_1$. We get $\varphi_e(0)\downarrow$; hence, $\varphi_{f(e)}$ is total as desired.

Suppose second $f(e) \in A_2$. Thus, $\varphi_{f(e)}(0)\downarrow$; hence, $\varphi_e(0)\downarrow$, which gives $f(e) \in A_1$ as desired.

All other proofs are left as exercises. □

Theorem 2.14. We have that

$$A_2 \not\leq_1 A_1.$$

Proof. Suppose, by way of contradiction, there is an $s \in \mathcal{R}$ such that, for all e,

$$\varphi_e \text{ is total } \Leftrightarrow \varphi_{s(e)}(0)\downarrow.$$

By **KRT** (Theorem 2.8), there is e such that, for all x,

$$\varphi_e(x) = \begin{cases} 0, & \text{if } \neg(\Phi_{s(e)}(0) \leq x) \\ \uparrow, & \text{otherwise.} \end{cases}$$

Clearly,

$$\varphi_e \text{ is total } \Leftrightarrow \varphi_{s(e)}(0)\uparrow,$$

a contradiction. □

Theorem 2.15. We have that

$$A_1 \not\leq_1 \overline{A_1} \text{ and } \overline{A_1} \not\leq_1 A_1.$$

Proof. Suppose, by way of contradiction, $A_1 \not\leq_1 \overline{A_1}$, as witnessed by $s \in \mathcal{R}$. Thus, for all e,

$$\varphi_e(0)\downarrow \Leftrightarrow \varphi_{s(e)}(0)\uparrow.$$

Using Rogers' Fixedpoint Theorem (Theorem 2.9), we see that there is an e such that $\varphi_e = \varphi_{s(e)}$, a contradiction.

Similarly, we get $\overline{A_1} \not\leq_1 A_1$. □

Theorem 2.16. For each p, let

$$\Theta(\varphi_p) = \begin{cases} \lambda x.0, & \text{if } \mathrm{dom}(\varphi_p) \text{ is infinite;} \\ \lambda x.\uparrow, & \text{otherwise.} \end{cases}$$

Then Θ is *not* an effective operator.

Proof. Suppose, by way of contradiction, Θ is an effective operator, as witnessed by $f \in \mathcal{R}$, i.e., for all p,

$$\varphi_{f(p)} = \begin{cases} \lambda x.0, & \text{if } \mathrm{dom}(\varphi_p) \text{ is infinite;} \\ \lambda x.\uparrow, & \text{otherwise.} \end{cases}$$

By **KRT** (Theorem 2.8), there is p such that, for all x,

$$\varphi_p(x) = \begin{cases} 0, & \text{if } \Phi_{f(p)}(0) \leq x; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Then we have

$$\mathrm{dom}(\varphi_p) \text{ is infinite; } \Leftrightarrow \varphi_{f(e)}(0)\uparrow,$$

a contradiction. □

Theorem 2.17. For each p, let

$$\Theta(\varphi_p) = \begin{cases} \lambda x.0, & \text{if } \mathrm{range}(\varphi_p) \text{ is infinite;} \\ \lambda x.\uparrow, & \text{otherwise.} \end{cases}$$

Then Θ is *not* an effective operator.

The proof of this theorem is left as an exercise.

Theorem 2.18. For all total computable f there is an e such that φ_e is extendable by a total computable function, but $\varphi_{f(e)}$ is not a total computable extension of φ_e.

Proof. Suppose, by way of contradiction, that we have an $f \in \mathcal{R}$ such that, for all e such that there is an $h \in \mathcal{R}$ with $\varphi_e \subseteq h$, $\varphi_e \subseteq \varphi_{f(e)} \in \mathcal{R}$.

By **KRT** (Theorem 2.8), there is e such that, for all x,

$$\varphi_e(x) = \varphi_{f(e)}(0) + 1.$$

Then we have that φ_e is either total or everywhere undefined; in either case, φ_e is extendable by a total computable function. Thus, by supposition, $\varphi_{f(e)}$ is total; hence, so is φ_e. However, $\varphi_{f(e)}(0) \neq \varphi_{f(e)}(0) + 1 = \varphi_e(0)$, a contradiction to $\varphi_{f(e)}$ extending φ_e. □

Theorem 2.19. For all total computable f there is an e such that the complement of $\mathrm{dom}(\varphi_e)$ contains *exactly* one element, but $\varphi_{f(e)}$ is not a total computable extension of φ_e.

The proof of this theorem is left as an exercise.

Theorem 2.20. Let

$$\mathcal{S}_0 = \{g \in \mathcal{R} \mid \varphi_{g(0)} = g\}.$$

Let $h \in \mathcal{R}$ be given. Then there is $g \in \mathcal{S}_0$ such that

$$\forall^\infty x : h(g[x]) \neq g(x).$$

Proof. By **KRT**, there is a e such that, for all x,

$$\varphi_e(z) = \begin{cases} e, & \text{if } x = 0; \\ h(\varphi_e[x]) + 1, & \text{otherwise.} \end{cases}$$

□

Applications of ORT In this section we give an application of ORT (Theorem 2.10); In particular, we can use ORT to show the existence of a padding function (Theorem 2.6). More theorems using ORT can be found in later sections on learning theory.

Theorem 2.21 (Padding Theorem)**.** There is a 1-1 total computable function pad such that, for all p, x,

$$\varphi_{\mathrm{pad}(p,x)} = \varphi_p.$$

Proof. By ORT, there is a function $f \in \mathcal{R}$ such that, for all x, y, z, $\varphi_{f(x,y)}(z) =$

$$\begin{cases} 0, & \text{if } f(x,y) \in \{f(0), \ldots, f(\langle x,y\rangle - 1)\}; \\ 1, & \text{else if } f(x,y) \in \{f(\langle x,y\rangle + 1), \ldots, f(z)\}; \\ \varphi_x(z), & \text{otherwise.} \end{cases}$$

It suffices to show that p is 1-1. Let x, x', y, y' be such that $f(x,y) = f(x',y')$. Suppose, by way of contradiction, $\langle x,y\rangle < \langle x',y'\rangle$. Then we have, for $z = \langle x',y'\rangle$,

$$1 = \varphi_{f(x,y)}(z) = \varphi_{f(x',y')}(z) = 0,$$

a contradiction. □

3 Basic Learning Theory

In this course we mainly study computational learning theory, specifically in the sense of *learning in the limit*; the most common name for this area of research is *inductive inference*, even though there are many fields of study concerned with this kind of inference.

This section discusses the basic notions of learning theory. We start by giving a brief overview over the history of learning theory, followed by the general idea of what is learning, and how does it relate to similar concepts of *identification* or *extrapolation* and so on.

3.1 History

The literature on inductive inference goes back to a paper by Gold [Gol67]. Gold successively did not publish many more papers, and it was not until Blum and Blum [BB75] that this line of research was picked up among western scientists. However, the mentioned paper by Gold by now got over 3000 citations (according to Google scholar).

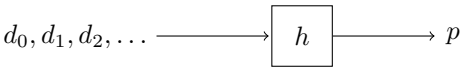
Meanwhile, the two Latvian researchers Rusins Freivalds and Janis Barzdins made valuable contributions starting in the 1970s to inductive inference. This influenced Rolf Wiehagen, a German researcher, who was very active in the area for a long period of time starting in the mid-1970s with many important contributions, inspiring colleagues and students and thus spreading the study of inductive inference.

In the western world, John Case picked up the paper by Blum and Blum and made successively many important contributions, thus spreading the study of inductive inference just as Rolf Wiehagen did.

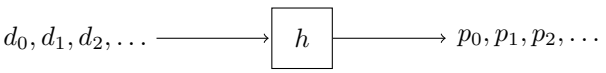
3.2 What is Learning?

In inductive inference, to *learn* means to derive, from a finite set of observations, knowledge about something unobserved. In this course, we will chiefly be concerned with deriving knowledge about *infinite sequences* by observing *finite parts* of these sequences.

One of the simplest models of learning is *finite learning*. In finite learning, the learner sees more and more data and, after finite time, makes an output. Learning is consider successful, if the output is a correct description of the input sequences (we consider each natural number p a correct description for φ_p).



The general setting we consider is as follows. A learner is presented with more and more of an unknown infinite sequence. In each step, the learner makes an output; the sequence of these outputs is the *learning sequence*. We have a relation on learning and input sequences to decide whether learning is successful.



We distinguish the following two kinds of learning: *online* learning, where a learner *extrapolates* finitely much at any point; and *off-line* learning, where a learner gives conjectures about the complete nature of the sequence to be learned.

Furthermore, we distinguish between cases of learnees that are static (or *passive*), and those that *react* to the learning process. Table 1 gives an overview of these settings.

In this course we will mostly consider the case of passive learnees.

	Passive Learnee	**Reactive Learnee**
Off-Line	Identification	Dynamic Modeling
On-Line	Extrapolation	Coordination

Table 1: The four categories of learning.

3.3 Formal Definitions

A *learner* is any partial computable function.

A *learnee* is any *total* computable function.

For this course, a *learning criterion* is a tuple $(\mathcal{C}, \alpha, \beta, \delta)$ as follows.

We call $\mathcal{C}$ the *learner admissibility restriction*; it can be any subset of $\mathcal{P}$ and intuitively defines what learners may be used for learning. An example is $\mathcal{C} = \mathcal{R}$, requiring all learners to be total.

We call α the *global learning restriction*; it defines a restriction on *all* learning sequences, even those on uninteresting data. An example is to require a learner to *always* output only programs for total functions.

We call β a *sequence generating operator*; β is a function taking a learner h and a learnee g and returns the *learning sequence* of h on g. Our standard example is the generation of learning sequences according to Gold's model [Gol67]; we denote his operator with $\mathbf{G}$ and define it as follows.

$$\mathbf{G} : (\mathcal{P} \times \mathcal{R}) \to \mathcal{P}; (h, g) \mapsto \lambda n . h(g[n]).$$

Intuitively, β defines how a learner processes a given learnee to produce a sequence of conjectures.

A *sequence acceptance criterion* is a predicate δ on a learning sequence p and a learnee g. We give the following examples.

$$\begin{aligned}
\mathbf{Ex}(p,g) &\Leftrightarrow [\forall^\infty i : \varphi_{p(i)} = g \wedge p(i) = p(i+1)];\\
\mathbf{Bc}(p,g) &\Leftrightarrow [\forall^\infty i : \varphi_{p(i)} = g];\\
\mathbf{Fin}(p,g) &\Leftrightarrow [\exists e : \mathrm{range}(p) \subseteq \{e, ?\} \wedge \varphi_e = g];\\
\mathbf{T}(p,g) &\Leftrightarrow [\forall i : \varphi_{p(i)} \in \mathcal{R}];\\
\mathbf{M}(p,g) &\Leftrightarrow [\forall^\infty i : p(i) = g(i)];\\
\mathcal{R}(p,g) &\Leftrightarrow p \in \mathcal{R}.
\end{aligned}$$

Note that any sequence acceptance criterion can also be used as a global learning restriction.

We combine any two sequence acceptance criteria δ and δ' by intersecting them, and we denote this combination by $\delta\delta'$.

Instead of writing the tuple $(\mathcal{C}, \alpha, \beta, \delta)$, we sometimes write $\mathcal{C}\tau(\alpha)\beta\delta$.

Definition 3.1. Let a learning criterion $I = (\mathcal{C}, \alpha, \beta, \delta)$ be given. For a learner h we say that h *I-learns* a learnee g iff

(1) $h \in \mathcal{C}$;

(2) for all $g' : \mathbb{N} \to \mathbb{N}$, with $p = \beta(h, g')$, $\alpha(p, g')$; and

(3) with $p = \beta(h, g)$, $\delta(p, g)$.

As discussed before, this means that $\mathcal{C}$ restricts the admissible learners, α gives a restriction that needs to hold for *all* possible learning sequences, and δ gives the restriction for the learning sequence resulting from h on g; learning sequences are defined by β.

For each learning criterion I, we denote the class of all total computable functions I-learned by h with $I(h)$. Abusing notation, we use $\mathcal{C}\tau(\alpha)\beta\delta$ to denote the set of all classes of languages I-learnable by some learner (as well as the learning criterion).

We omit $\tau(\alpha)$ if α gives no restriction; similarly with $\mathcal{C} = \mathcal{P}$.

3.4 Examples

In this section we present a number of example learning criteria that can be built from the definitions of Section 3.3.

Finite learning, from Section 3.2, is captured by **GFin**; the learner can look at more and more data, and, eventually, output something different from ?, its final answer.

Gold's first learning criterion, learning as identification in the limit, is given by **GEx**. In particular, a learner h **GEx**-learns a learnee g iff there is i_0 such that, for all $i \geq i_0$, $h(g[i]) = h(g[i_0])$ and $\varphi_{h(g[i_0])} = g$. Intuitively, the learner finds an *explanation* for the data given.

An example of *extrapolation* is given by **GM**. In particular, a learner h **GM**-learns a learnee g iff there is i_0 such that, for all $i \geq i_0$, $h(g[i]) = g(i)$. Intuitively, the learner h extrapolates the sequence one element at a time.

4 Extrapolation and Exact Identification

In this section we give some theorems regarding learning criteria. Of some importance throughout learning theory are the following two sets of total computable functions. Firstly, the set

$$\mathcal{S}_{\text{FinSup}} = \{g \in \mathcal{R} \mid \forall^\infty x : g(x) = 0\}$$

is the set of all functions of *finite support*, i.e., functions which are only non-0 on finitely many inputs. Secondly, the set

$$\mathcal{S}_{\text{SD}} = \{g \in \mathcal{R} \mid \varphi_{g(0)} = g\}$$

is the set of all *self-describing* functions, i.e., functions, which give a description (in the sense of φ-programs) of themselves on input 0.

We start with some theorems on extrapolation and exact identification, also relating these two concepts.

The first theorem we give deals with different kinds of extrapolation. The main model of extrapolation is given by the learning criterion **GM**. Note that in this model, the learner may diverge on finitely much input even for target learnees; with $\mathbf{G}\mathcal{R}\mathbf{M}$ we can model the additional restriction that, at least on target learnees, the learner has to be required to be total. Finally, $\mathcal{R}\mathbf{GM}$ models extrapolation by total learners. The following theorem shows that each successive restriction lowers the learning power in these criteria.

Theorem 4.1. We have

$$\mathcal{R}\mathbf{GM} \subset \mathbf{G}\mathcal{R}\mathbf{M} \subset \mathbf{GM}.$$

Proof. Each inclusion is trivial, as more left criteria are more restricted then more right criteria. It remains to show that the learning criteria are indeed different in learning power.

We start by showing $\mathcal{R}\mathbf{GM} \neq \mathbf{G}\mathcal{R}\mathbf{M}$. We do this by first showing that $\mathcal{S}_{\text{SD}} \in \mathbf{G}\mathcal{R}\mathbf{M}$; then we show that $\mathcal{S}_{\text{SD}} \notin \mathcal{R}\mathbf{GM}$.

Let $h \in \mathcal{P}$ be defined such that, for all σ,

$$h(\sigma) = \begin{cases} 0, & \text{if } \sigma = \emptyset; \\ \varphi_{\sigma(0)}(\text{len}(\sigma)), & \text{otherwise.} \end{cases}$$

Let $g \in \mathcal{S}_{\text{SD}}$. Then we have $h(g[0])\downarrow$, as well as, for all $i > 0$,

$$h(g[i]) = \varphi_{g(0)}(i) = g(i).$$

This shows that h on g always converges and almost always gives the correct extrapolation.

To show that $\mathcal{S}_{\text{SD}} \notin \mathcal{R}\mathbf{GM}$ we suppose, by way of contradiction, otherwise, as witnessed by some $h \in \mathcal{R}$. By **KRT**, there is a e such that, for all x,

$$\varphi_e(x) = \begin{cases} e, & \text{if } x = 0; \\ h(\varphi_e[x]) + 1, & \text{otherwise.} \end{cases}$$

Let $g = \varphi_e$. Clearly, $g \in \mathcal{S}_{\text{SD}}$. Now we have, for all $i > 0$,

$$g(i) = h(g[i]) + 1 \neq h(g[x]).$$

Next, we show $\mathbf{G}\mathcal{R}\mathbf{M} \neq \mathbf{GM}$ by showing that the set

$$\mathcal{S}_0 = \{g \in \mathcal{R} \mid \forall^\infty n : \varphi_{g(n)} = g\}$$

serves as a separator. Let $h \in \mathcal{P}$ be defined such that, for all σ,

$$h(\sigma) = \begin{cases} 0, & \text{if } \sigma = \emptyset; \\ \varphi_{\text{last}(\sigma)}(\text{len}(\sigma)), & \text{otherwise.} \end{cases}$$

Let $g \in \mathcal{S}_0$. Then we have, for all i large enough,

$$h(g[i]) = \varphi_{g(i-1)}(i) = g(i).$$

This shows that h on g almost always gives the correct extrapolation.

To show that $\mathcal{S}_0 \notin \mathbf{G}\mathcal{R}\mathbf{M}$ we first show that $\mathcal{S}_0$ is *dense*.[2] To that end, let σ be a sequence. By **KRT**, there is a e such that, for all x,

$$\varphi_e(x) = \begin{cases} \sigma(x), & \text{if } x < \text{len}(\sigma); \\ e, & \text{otherwise.} \end{cases}$$

Clearly, $g \in \mathcal{S}_0$ and $\sigma \subseteq g$. From $\mathcal{S}_0$ being dense, we see that h has to be defined on all inputs (since all inputs can be extended to something that h learns, and h is always defined on something that it learns); i.e., $h \in \mathcal{R}$.

Now we get, using **KRT**, that there is a e such that, for all x,

$$\varphi_e(x) = \begin{cases} e, & \text{if } x = 0; \\ \text{pad}(e, h(\varphi_e[x]) + 1), & \text{otherwise.} \end{cases}$$

Let $g = \varphi_e$. Clearly, $g \in \mathcal{S}_{\text{SD}}$. Now we have, for all $i > 0$,

$$g(i) = h(g[i]) + 1 \neq h(g[x]).$$

□

In contrast to the results regarding extrapolation, we have the following result on total learners in identification. We call a sequence acceptance criterion δ *delayable* iff, for all *monotone* functions r such that

- $\lim_{x\to\infty} r(x) = \infty$ and
- $\forall x : r(x) \leq x$,

we have that with any $(p, g) \in \delta$ we get $(p \circ r, g) \in \delta$. Many δ, such as **Ex**, **Bc** and **T** are delayable, but not **M**. We get the following theorem.

Theorem 4.2. Let δ be delayable. Then

$$\mathbf{G}\delta = \mathcal{R}\mathbf{G}\delta.$$

In fact, all learners can be assumed *linear time* computable.

Proof.

□

The next theorem shows that there is no omnipotent **GEx**-learner.

Theorem 4.3. We have

$$\mathcal{R} \notin \mathbf{GEx}.$$

Proof. Suppose, by way of contradiction, there is $h \in \mathcal{P}$ such that $\mathcal{R} = \mathbf{GEx}(h)$. We first show the following claim.

$$\forall \sigma \exists \tau : h(\sigma \diamond \tau) \neq h(\sigma). \tag{3}$$

Intuitively, h makes a mind change *somewhere* after any arbitrary σ. Let σ be given, and let, for each $j \in \{0, 1\}$, $g_j \in \mathcal{R}$ be the extension of σ with infinitely many js. Then h **GEx**-learns both g_0 and g_1; thus, there are $t_0, t_1 > \text{len}(\sigma)$ such that

$$\varphi_{h(g_0[t_0])} = g_0 \text{ and } \varphi_{h(g_1[t_1])} = g_1.$$

Note that we can choose t_0 and t_1 to be larger than any number we want (for example $\text{len}(\sigma)$), since h has to make a correct output infinitely often.

Thus, we get $h(g_0[t_0]) \neq h(g_1[t_1])$, so one of the two extensions $g_0[t_0]$ and $g_0[t_0]$ of σ will serve as the τ required for (3).

Now we inductively define a (computable) family of sequences $(\sigma_i)_{i\in\mathbb{N}}$ as follows.

$$\begin{aligned} \sigma_0 &= \emptyset; \\ \forall i : \sigma_{i+1} &= \sigma_i \diamond \mu\tau . h(\sigma_i \diamond \tau) \neq h(\sigma_i). \end{aligned}$$

By (3), for all i, σ_i is defined. Let $g = \bigcup_{i\in\mathbb{N}} \sigma_i$. Then $g \in \mathcal{R}$, but h makes infinitely many mind changes on g, a contradiction. □

The learning criterion **GBc** is much more powerful than **GEx**. The following theorem gives the separation.

Theorem 4.4. We have

$$\mathbf{GEx} \subset \mathbf{GBc}.$$

Proof. We use the set

$$\mathcal{S}_0 = \{g \in \mathcal{R} \mid \forall^\infty n : \varphi_{g(n)} = g\}$$

just as in Theorem 4.1. Clearly, we have $\mathcal{S}_0 \in \mathbf{GBc}$ as witnessed by $h \in \mathcal{P}$ such that

$$\forall \sigma : h(\sigma) = \begin{cases} 0, & \text{if } \sigma = \emptyset; \\ \text{last}(\sigma), & \text{otherwise.} \end{cases}$$

To show that $\mathcal{S}_0 \notin \mathbf{GEx}$, suppose, by way of contradiction, there is $h \in \mathcal{R}$ such that $\mathcal{S}_0 \in \mathbf{GEx}(h)$.

By ORT, there are a computable family of finite sequences $(\sigma_i)_{i\in\mathbb{N}}$, $e \in \mathbb{N}$ and a 1-1 $f \in \mathcal{R}$ such that, for all i, j, x, σ,[3]

$$\begin{aligned} \sigma_0 &= \emptyset; \\ \sigma_{i+1} &= \sigma_i \diamond \mu\tau . \begin{cases} \tau \in \{f(\sigma_i, 0)^t, f(\sigma_i, 1)^t \mid t \in \mathbb{N}\} \wedge \\ h(\sigma_i \diamond \tau) \neq h(\sigma_i); \end{cases} \\ \varphi_e(x) &= \bigcup_{i\in\mathbb{N}} \sigma_i; \\ \varphi_{f(\sigma,j)}(x) &= \begin{cases} \sigma(x), & \text{if } x < \text{len}(\sigma); \\ f(\sigma, j), & \text{if } \forall y, \text{len}(\sigma) < y < x : \\ & \quad h(\varphi_{f(\sigma,j)}[y]) = h(\sigma); \\ \varphi_e(x), & \text{otherwise.} \end{cases} \end{aligned}$$

[2] A set $\mathcal{S} \subseteq \mathcal{R}$ is called *dense* iff, for all sequences σ, there is $g \in \mathcal{S}$ with $\sigma \subseteq g$.

[3] In the minimization over a set of sequences, we suppose that the coding of sequences is monotone with respect to sequence extensions, i.e., if σ is a prefix of τ, then the code of σ is numerically less than the code of τ.

We first show by induction that, for each i, σ_i is defined, with trivial induction begin. Let $i \in \mathbb{N}$ be such that σ_i is defined. Suppose, by way of contradiction, σ_{i+1} is not defined. Let, for each $j \in \{0,1\}$, $g_j \in \mathcal{R}$ be the extension of σ_i with infinitely many $f(\sigma_i, j)$s. It is easy to see that now we have, for each $j \in \{0,1\}$,

$$g_j = \varphi_{f(\sigma_i,j)}$$

and $g_j \in \mathcal{S}_0$. Thus, h **GEx**-learns both g_0 and g_1, which shows that h needs to make a mind change on one of them, similarly to Theorem 4.3, a contradiction.

Thus, we get that φ_e is total; let $g = \varphi_e$. We have that h makes infinitely mind changes on g, so it suffices to show that $g \in \mathcal{S}_0$. Let $n \in \mathbb{N}$, and suppose that, for some i, j, $g(n) = f(\sigma_i, j)$. Thus, with t minimal such that

$$h(\sigma_i \diamond f(\sigma_i, j)^t) \neq h(\sigma_i), \tag{4}$$

we have

$$\sigma_{i+1} = \sigma_i \diamond f(\sigma_i, j)^t. \tag{5}$$

For all x we have $\varphi_{g(n)}(x) =$

$$\varphi_{f(\sigma_i,j)}(x) = \begin{cases} \sigma_i(x), & \text{if } x < \operatorname{len}(\sigma_i); \\ f(\sigma_i, j), & \text{if } \forall y, \operatorname{len}(\sigma_i) < y < x : \\ & \quad h(\varphi_{f(\sigma_i,j)}[y]) = h(\sigma_i); \\ \varphi_e(x), & \text{otherwise.} \end{cases}$$

In the first and the last case, we immediately get

$$\varphi_{g(n)}(x) = g(x)$$

as desired. As t is minimal such that (4) holds, we have that the middle case holds iff $\operatorname{len}(\sigma_i) \leq x \leq t$, in which case $x \leq \sigma_{i+1}$ by (5). Thus, for all such x,

$$\varphi_{g(n)}(x) = f(\sigma_i, j) = \sigma_{i+1}(x) = g(x).$$

This finishes the proof. □

As powerful as **GBc** is, it does not encompass all unions of **GEx**-learnable sets, as shown by the following theorem.

Theorem 4.5. We have $\mathcal{S}_{\text{FinSup}}, \mathcal{S}_{\text{SD}} \in \mathbf{GEx}$, but

$$\mathcal{S}_{\text{FinSup}} \cup \mathcal{S}_{\text{SD}} \notin \mathbf{GBc}.$$

This entails $\mathcal{R} \notin \mathbf{GBc}$.

Proof. Similarly as in Theorem 2.12, there is a function $r \in \mathcal{R}$ (using the S-m-n Theorem) such that, for all σ,

$$\forall x : \varphi_{r(\sigma)}(x) = \begin{cases} \sigma(x), & \text{if } x < \operatorname{len}(\sigma); \\ 0, & \text{otherwise.} \end{cases}$$

For each finite sequence σ, let $\hat{\sigma}$ denote σ with trailing 0s removed. Let $h \in \mathcal{P}$ be such that, for all σ,

$$h(\sigma) = r(\hat{\sigma}).$$

It is now easy to verify that h **GEx**-learns $\mathcal{S}_{\text{FinSup}}$.

Let $h' \in \mathcal{P}$ be such that, for all σ,

$$h'(\sigma) = \begin{cases} ?, & \text{if } \sigma = \emptyset; \\ \sigma(0), & \text{otherwise.} \end{cases}$$

Clearly, h' even shows $\mathcal{S}_{\text{SD}} \in \mathbf{GFin}$.

An easier version of the claim "$\mathcal{S}_{\text{FinSup}} \cup \mathcal{S}_{\text{SD}} \notin \mathbf{GBc}$" is left as an exercise, so this will claim will not be shown before this exercise is due. □

What can be learned? The following theorem gives a large class of sets of functions which can be learned, namely all uniformly computable sets of functions. This is called "learning by enumerating."

Theorem 4.6. Let $f \in \mathcal{R}$. Let $\mathcal{S} = \{\lambda x.f(e,x) \mid e \in \mathbb{N}\}$. Then

$$\mathcal{S} \in \tau(\mathbf{T})\mathbf{GEx}.$$

The proof of this theorem is left as an exercise.

Let $\mathbb{U}\text{Comp}$ be the set of all classes $\mathcal{S}$ such that there is an f with $\mathcal{S} = \{\lambda x.f(e,x) \mid e \in \mathbb{N}\}$; these sets are called *sets of uniformly computable function*. We get the following theorem about total learning.

Theorem 4.7. We have

$$\{\mathcal{S} \subseteq \mathcal{R} \mid \exists \mathcal{S}' \in \mathbb{U}\text{Comp} : \mathcal{S} \subseteq \mathcal{S}'\} = \tau(\mathbf{T})\mathbf{GEx}$$

and

$$\tau(\mathbf{T})\mathbf{GEx} \subset \mathbf{GTEx} \subset \mathbf{GEx}.$$

Proof. The inclusion $\mathbb{U}\text{Comp} \subseteq \tau(\mathbf{T})\mathbf{GEx}$ was shown in Theorem 4.6. The converse inclusion follows from the observation, that any learner h fulfilling $\tau(\mathbf{T})$ has a range of only total functions, so the range h would be programs for a set of uniformly computable functions.

Next we show $\mathcal{S}_{\text{SD}} \in \mathbf{GTEx} \setminus \tau(\mathbf{T})\mathbf{GEx}$. The inclusion is trivial. Suppose, by way of contradiction, there is $h \in \mathcal{R}$ such that $\mathcal{S}_{\text{SD}} \subseteq \tau(\mathbf{T})\mathbf{GEx}(h)$. By **KRT**, there is e such that

$$\forall x : \varphi_e(x) = \begin{cases} e, & \text{if } x = 0; \\ \varphi_{h(\varphi_e[x])} + 1, & \text{otherwise.} \end{cases}$$

We have that φ_e is total, as h only outputs programs for total functions; thus, $\varphi_e \in \mathcal{S}_{\text{SD}}$. Clearly, h does not **GEx**-learn φ_e, a contradiction.

Finally, we show $\mathbf{GTEx} \subset \mathbf{GEx}$. We give two proofs for this, one with a hand-crafted set $\mathcal{S}$, and one with a *self-learning set of functions*.

Proof 1: Let

$$\mathcal{S} = \{g \in \mathcal{R} \mid \exists e \forall^\infty n : \pi_1(g(n)) = e \wedge \varphi_e = g\}.$$[4]

[4] Recall that π_1 is a functions such that, for all x, y, $\pi_1(\langle x, y\rangle) = x$, i.e., π_1 extracts the first component from a pair.

Observe that $\mathcal{S}$ is dense, and, clearly, $\mathcal{S} \in \mathbf{GEx}$. We now show $\mathcal{S} \notin \mathbf{GTEx}$. Suppose, by way of contradiction, otherwise, as witnessed by $h' \in \mathcal{P}$. As $\mathcal{S}$ is dense, we have $h' \in \mathcal{R}$; furthermore, also from denseness, h' outputs programs for total functions on all inputs.

By **KRT**, there is e such that, for all x,

$$\varphi_e(x) = \mu y . y \in \{\langle e,0\rangle, \langle e,1\rangle\} \setminus \{\varphi_{h'(\varphi_e[x])}(x)\}.$$

As h makes only outputs for total functions, we get that φ_e total; thus, $\varphi_e \in \mathcal{S}$. But of course h' does not **GTEx**-identify φ_e, a contradiction.

Proof 2: Let h be such that, for all $\sigma \neq \emptyset$,

$$h(\sigma) = \varphi_{\mathrm{last}(\sigma)}(\sigma).$$

Let $\mathcal{S} = \mathbf{GEx}(h)$; $\mathcal{S}$ is called a self-learning set of functions, as h does not do much work – the main parts of identification are done by running the programs provided by the functions from the sets themselves.

Clearly, $\mathcal{S} \in \mathbf{GEx}$ (by definition). Observe that $\mathcal{S}$ is dense. We now show $\mathcal{S} \notin \mathbf{GTEx}$. Suppose, by way of contradiction, otherwise, as witnessed by $h' \in \mathcal{P}$. As $\mathcal{S}$ is dense, we have $h' \in \mathcal{R}$.

Let $r \in \mathcal{R}$ be strictly monotone increasing such that, for all e, x, y,

$$\varphi_{r(e,x)}(y) = e.$$

By **KRT**, there is e such that, for all x,

$$\varphi_e(x) = r(e, \varphi_{h'(\varphi_e[x])}(x) + 1).$$

From $\mathcal{S}$ dense, we get that h makes only outputs for total functions. This gives φ_e total; thus, $\varphi_e \in \mathcal{S}$. But of course h' does not **GTEx**-identify φ_e, a contradiction. □

The following theorem relates the learning power of extrapolation and identification.

Theorem 4.8. We have the following.

(1) $\mathcal{R}\mathbf{GM} = \tau(\mathbf{T})\mathbf{GEx}$;

(2) $\mathbf{GTEx} \subset \mathbf{G}\mathcal{R}\mathbf{M} \subset \mathbf{GEx}$;

(3) $\mathbf{GM} = \mathbf{GBc}$.

Proof. We show only item 3., leaving the others as an exercise. Let $h \in \mathcal{P}$ be given. We view h as a trying to identify a set of functions an will now construct h' to extrapolate the same set of functions. Let $h' \in \mathcal{P}$ be such that, for all σ,

$$h'(\sigma) = \varphi_{h(\sigma)}(\mathrm{len}(\sigma)).$$

Let $g \in \mathbf{GBc}(h)$; we will show that $g \in \mathbf{GM}(h')$. Let t_0 be such that, for all $t \geq t_0$, $\varphi_{h(g[t])} = g$. Thus we have, for all $t \geq t_0$,

$$h'(g[t]) = \varphi_{h(g[t])}(t) = g(t).$$

This shows that g is **GM**-learned (extrapolated) by h'.

Consider now $h \in \mathcal{P}$ as an extrapolator. By ORT, there is a function $h' \in \mathcal{R}$ such that, for all σ, x,

$$\varphi_{h'(\sigma)}(x) = \begin{cases} \sigma(x), & \text{if } x < \mathrm{len}(\sigma); \\ h(\varphi_{h'(\sigma)}[x]), & \text{otherwise.} \end{cases}$$

Let $g \in \mathbf{GM}(h)$; we show that $g \in \mathbf{GBc}(h')$. Let t_0 be such that, for all $t \geq t_0$, $h(g[t]) = g(t)$ and let $t \geq t_0$. We show by induction on x that $\varphi_{h'(g[t])}(x) = g(x)$; this is trivial for $x < t$. Let now $x \geq t$ be such that, for all $y < x$, $\varphi_{h'(g[t])}(y) = g(y)$; thus, $\varphi_{h'(g[t])}[x] = g[x]$. We now have

$$\varphi_{h'(g[t])}(x) = h(\varphi_{h'(g[t])}[x]) = h(g[x]) = g(x).$$

This finishes the induction. Therefore, $g \in \mathbf{GBc}(h')$ as desired.

We now show $\mathbf{GTEx} \neq \mathbf{G}\mathcal{R}\mathbf{M}$. Let

$$\mathcal{S} = \{g \in \mathcal{R} \mid \forall n : \varphi_{g(n)}(0)\!\downarrow \wedge \forall^\infty n : \varphi_{g(n)}(0) = g(n+1)\}.$$

Clearly, $\mathcal{S}$ is $\mathbf{G}\mathcal{R}\mathbf{M}$-learnable by $\lambda\sigma . \varphi_{\mathrm{last}(\sigma)}(0)$. Suppose, by way of contradiction, $\mathcal{S} \in \mathbf{GTEx}$, as witnessed by $h \in \mathcal{R}$. By ORT, there is e and a strictly monotone increasing $f \in \mathcal{R}$ such that, with $g = \varphi_e$, for all x, g(x) =

$$\begin{cases} f(x,0), & \text{if } x < 2; \\ \min(\{f(x,0), f(x,1)\} \setminus \{\varphi_{h(g[x-2])}(x) + 1\}) & \text{otherwise;} \end{cases}$$

and, for all n, i, x,

$$\varphi_{f(n,i)}(x) = g(n+1).$$

We will show, by induction on n, that for all n, $g(n)\!\downarrow$ and $\varphi_{g(n)}(0)\!\downarrow$. This will prove the claim, as this gives $g \in \mathcal{S}$ and h does not **GTEx**-learn g.

Let $n > 1$ be such that, for all $i < n$, $g(i)\!\downarrow$ and $\varphi_{g(n)}(0)\!\downarrow$. By ORT, there are p, q such that

$$\begin{aligned} \varphi_p(x) &= g[n-1] \diamond q^\infty \\ \varphi_q(x) &= q. \end{aligned}$$

Using the induction hypothesis we get $\varphi_p \in \mathcal{S}$. Thus, $\varphi_{h(g[n-2])}$ and $\varphi_{h(g[n-1])}$ are, by assumption on h, total. This gives $g(n)\!\downarrow$ and $g(n+1)\!\downarrow$; in particular, $\varphi_{g(n)}(0) = g(n+1)\!\downarrow$, which concludes the induction and, thus, the proof. □

As the last theorem in this section we prove that $\mathcal{R}$ is not **GBc**-learnable; this gives a first example for how to prove that some set of functions is not **GBc**-learnable.

Theorem 4.9. We have

$$\mathcal{R} \notin \mathbf{GBc}.$$

Proof. Suppose, by way of contradiction, there is $h \in \mathcal{R}$ such that $\mathcal{R} = \mathbf{GBc}(h)$. We first show the following claim.

$$\forall \sigma \exists \tau, y : \varphi_{h(\sigma \diamond \tau)}(y)\downarrow \wedge y \geq \mathrm{len}(\sigma \diamond \tau). \tag{6}$$

Intuitively, we show that we can extend any σ such that h makes an extrapolation.

Let σ be given, and let $g_0 \in \mathcal{R}$ be the extension of σ with infinitely many 0s. Then h **GBc**-learns g_0 ; thus, there are $t_0 > \mathrm{len}(\sigma)$ such that

$$\varphi_{h(g_0[t_0])} = g_0.$$

This shows Equation (3).

Now we inductively define a (computable) family of sequences $(\sigma_i)_{i \in \mathbb{N}}$ as follows. We let σ_0 be $\emptyset$. If σ_i is already defined, then we search for an extension τ and a y as in Equation (3). Then we concatenate σ with τ and fill up the resulting sequence with elements different from $\varphi_{h(\sigma \diamond \tau)}(y)$; this is our σ_{i+1}. Using Equation (3), we get $g \in \mathcal{R}$, but h is infinitely wrong on g (outputs a program for a function not correctly computing g), a contradiction. □

5 Almost Everywhere Correct Learning

In this section we consider learning with (finitely many) errors. Instead of requiring to find a program that correctly computes the input function on *all* arguments, we may be happy programs that are correct only on all but finitely many arguments.

Definition 5.1. Let $n \in \mathbb{N}$ and f, f' be functions. We write $f =^n f'$ iff f and f' are the same function on all but *at most* n arguments. Furthermore, we write $f =^* f'$ iff the set of arguments on which f and f' differ is *finite*. We use $*$ as a special symbol and write, for all $n \in \mathbb{N}$, $n < *$.

Let $a \in \mathbb{N} \cup \{*\}$. We define the following sequence acceptance criteria on a learning sequence p and a learnee g.

$$\begin{aligned} \mathbf{Ex}^a(p,g) &\Leftrightarrow [\forall^\infty i : \varphi_{p(i)} =^a g \wedge p(i) = p(i+1)]; \\ \mathbf{Bc}^a(p,g) &\Leftrightarrow [\forall^\infty i : \varphi_{p(i)} =^a g]. \end{aligned}$$

Not surprisingly, criteria based on such inexact identification is more powerful than exact identification. The next theorems deal with proving this, culminating in the surprising result that $\mathbf{GBc}^*$ learning is maximally powerful, i.e., there is a $\mathbf{GBc}^*$-learner which learns *each* total computable function (see Theorem 5.5).

But first we show how the hierarchies of learning classes for errors with **Ex**-learning, with **Bc**-learning, and how these hierarchies relate.

Theorem 5.2. Let $a, b \in \mathbb{N} \cup \{*\}$ with $a < b$. Then we have

$$\mathbf{GEx}^a \subset \mathbf{GEx}^b.$$

This proof is left as an exercise.

Theorem 5.3. Let $a, b \in \mathbb{N} \cup \{*\}$ with $a < b$. Then we have

$$\mathbf{GBc}^a \subset \mathbf{GBc}^b.$$

Proof. We only show $\mathbf{GBc}^0 \subset \mathbf{GBc}^1$. Let

$$\mathcal{S} = \{g \in \mathcal{R} \mid \forall^\infty n : \varphi_{g(n)} =^1 g\}.$$

Using $\lambda \sigma . \mathrm{last}(\sigma)$, we see that $\mathcal{S} \in \mathbf{GBc}^1$. Suppose, by way of contradiction, $\mathcal{S} \in \mathbf{GBc}^0$ as witnessed by $h \in \mathcal{R}$; fix a program for h.

By ORT, there are a computable family of finite sequences $(\sigma_i)_{i \in \mathbb{N}}$, $e \in \mathbb{N}$ and a 1-1 $P, f_1, f_2, p \in \mathcal{R}$ such that, for all i, j, x, t, σ,

$$\begin{aligned} P(\sigma,j,t) &= \varphi_{h(\sigma \diamond p(\sigma,j) \diamond p(\sigma,0)^t}(\mathrm{len}(\sigma))\downarrow \neq p(\sigma,j); \\ f_1(\sigma) &= \pi_1(\mu \langle j,t,s \rangle . P(\sigma,j,t)\downarrow_s) \\ f_2(\sigma) &= \pi_2(\mu \langle j,t,s \rangle . P(\sigma,j,t)\downarrow_s) \\ \sigma_0 &= \emptyset; \\ \sigma_{i+1} &= \sigma_i \diamond p(\sigma_i, f_1(\sigma)) \diamond p(\sigma_i,0)^{f_2(\sigma)}; \\ \varphi_e(x) &= \bigcup_{i \in \mathbb{N}} \sigma_i; \\ \varphi_{p(\sigma,j)}(x) &= \begin{cases} \sigma(x), & \text{if } x < \mathrm{len}(\sigma); \\ p(\sigma, f_1(\sigma)), & \text{else if } x = \mathrm{len}(\sigma); \\ p(\sigma,0), & \text{else if } f_2(\sigma)\uparrow; \\ p(\sigma,0), & \text{else if } x - \mathrm{len}(\sigma) \leq f_2(\sigma); \\ \varphi_e(x), & \text{otherwise.} \end{cases} \end{aligned}$$

□

Theorem 5.4. We have

$$\mathbf{GEx}^* \subset \mathbf{GBc}.$$

Proof. We show "$\subseteq$" and leave the separation as an exercise. Let $\mathcal{S} \in \mathbf{GEx}^*$ as witnessed by $h \in \mathcal{R}$.

As in Theorem 2.12, let $p \in \mathcal{R}$ be such that, for all finite sequences of natural numbers σ and e, x,

$$\varphi_{p(\sigma,e)}(x) = \begin{cases} \sigma(x), & \text{if } x < \mathrm{len}(\sigma); \\ \varphi_e(x), & \text{otherwise.} \end{cases}$$

Let $h' \in \mathcal{R}$ be such that, for all σ, $h'(\sigma) = p(h(\sigma))$.

Let $g \in \mathcal{S}$ and let t_0 be such that, for all $t \geq t_0$, $h(g[t]) = h(g[t_0])$ is a correct program for g, up to finitely many mistakes. Let $t_1 \geq t_0$ be such that, for all $t \geq t$, $h(g[t_0])$ is a correct program for all arguments $t \geq t_1$. Thus, for all $t \geq t_1$, $h'(g[t]) = p(h(g[t]))$ is a correct program for g. □

Theorem 5.5. We have

$$\mathcal{R} \in \mathbf{GBc}^*.$$

Proof. Using ORT, there is $h \in \mathcal{R}$ such that, for all σ, x,

$$\varphi_{h(\sigma)}(x) = \begin{cases} \varphi_p(x), & \text{if } p \leq \text{len}(\sigma) \text{ is the least number s.t.} \\ & \forall y < \text{len}(\sigma) : \Phi_p(y) \leq x \wedge \\ & \varphi_p(x) = \sigma(x); \\ 0, & \text{if no such } p \text{ exists.} \end{cases}$$

Let $g \in \mathcal{R}$ and let p be the least index for g. Let t_0 be large enough such that $p \leq t_0$ and, for all $p' < p$, $\varphi_{p'}[t_0] \neq g[t_0]$.

We show that, for all $t \geq t_0$, $\varphi_{h(g[t])} =^* g$. Let $t \geq t_0$ and let $x_0 = \max\{\Phi_p(y) \mid y < t\}$. We have, for all $x \geq x_0$,

$$\varphi_{h(g[t])}(x) = \varphi_p(x),$$

as t is large enough so that h on $g[t]$ considers p, while t is also large enough so that all $p' < p$ are either not convergent or incorrect; finally, we also have that x is large enough so that p is convergent on all necessary inputs.

This shows that h $\mathbf{GBc}^*$-identifies g. □

6 Consistency and Iterativeness

In this section, we first introduce *consistency* (and a somewhat weaker variant, *conformality*), followed by the notion of *iterative* learning. Then we relate these notions.

Consistency is the requirement that a learner is correct on all data it has seen so far. Formally, we model this restriction as a sequence acceptance criterion **Cons** such that

$$\mathbf{Cons} = \{(p, g) \mid \forall n : p(n) =? \vee \forall x < n : \varphi_{p(n)}(x) = g(x)\}.$$

In other words, when some information $g[n]$ is available, then the conjecture is for a function which starts with $g[n]$.

Conformality, on the other hand, allows for undefined values in $\varphi_{p(n)}$ – only *contradictory* output is forbidden. Formally, we define sequence acceptance criterion **Conf** such that

$$\mathbf{Conf} = \{(p, g) \mid \forall n \forall x < n : \varphi_{p(n)}(x)\!\downarrow \neq ? \Rightarrow \varphi_{p(n)}(x) = g(x)\}.$$

Iterative learning requires the learner to forget all old data, and only react to the current, newest datum. In addition, the learner is allowed access to its current hypothesis. Formally, we model iterative learning as a sequence generating operator **It** defined recursively as follows. For a given learner h and learnee g,

$$\begin{aligned} \mathbf{It}(h, g)(0) &= ?; \\ \forall n : \mathbf{It}(h, g)(n+1) &= h(\mathbf{It}(h, g)(n), n, g(n)). \end{aligned}$$

Note that the new datum is, implicitly, $\langle n, g(n)\rangle$, i.e., the learner gets to know *where* in the sequence the given function value is.

Theorem 6.1. We have

$$\begin{aligned} \tau(\mathbf{Cons})\mathbf{GEx} &= \tau(\mathbf{Conf})\mathbf{GEx} \subset \mathbf{GConsEx} \\ &\subset \mathbf{ItEx} \subset \mathbf{GEx}. \end{aligned}$$

Proof. Regarding "$\tau(\mathbf{Conf})\mathbf{GEx} \subseteq \tau(\mathbf{Cons})\mathbf{GEx}$", let $\mathcal{S} \in \tau(\mathbf{Conf})\mathbf{GEx}$ as witnessed by h.

Let $g \in \mathcal{P}$ be such that, for all σ, x,[5]

$$g_\sigma(x) = \begin{cases} \sigma(x), & \text{if } x < \text{len}(\sigma); \\ \mu y . h(g_\sigma[x] \diamond y) = h(g_\sigma[x]), & \text{otherwise.} \end{cases}$$

With s-m-n, we let $p \in \mathcal{R}$ be such that, for all σ, $\varphi_{p(\sigma)} = g_\sigma$.

We define h' for all σ, x such that

$$h'(\sigma \diamond x) = \begin{cases} h'(\sigma), & \text{if } \forall y < x : h(\sigma \diamond y) \neq h(\sigma) \wedge \\ & \quad h(\sigma \diamond x) = h(\sigma); \\ p(\sigma \diamond x), & \text{otherwise.} \end{cases}$$

We first show that, for all σ, $\sigma \subseteq \varphi_{h'(\sigma)}$. Let σ be a sequence. Let $\sigma' \subseteq \sigma \diamond x$ be such that $h'(\sigma \diamond x) = p(\sigma')$. We show, by induction on n,

$$\forall n \leq \text{len}(\sigma) : \varphi_{p(\sigma')}(n) = (\sigma \diamond x)(n).$$

This is clear for all $n < \text{len}(\sigma')$. Suppose now the claim holds for some n with $\text{len}(\sigma') < n < \text{len}(\sigma)$. By the definition of σ', we have that for h' on $(\sigma \diamond x)[n+1]$, the first case holds. Thus, using the induction hypothesis, for $\varphi_{p(\sigma')}$ on $n+1$ we have that the second case holds and evaluates to $(\sigma \diamond x)(n+1)$ as desired.

Furthermore, for any g, note that h' on g only changes its mind when h does, or when alternate data leads to the same conjecture. However, alternate data can only lead to the same conjecture in a conformal learner, if the conjecture is not total; in particular, the conjecture cannot be correct. In other words, once enough data of g is available such that h has converged to the correct conjecture, this conjecture will be for a total function, and the second case in the definition of h' cannot hold any more (which gives that h' has also converged). Similarly, it is easy to see that after the convergence of h to a correct conjecture, all conjectures of h' are also correct.

Regarding "$\tau(\mathbf{Cons})\mathbf{GEx} \subset \mathbf{GConsEx}$", we see that $\mathcal{S}_{\text{SD}}$ is trivially **GConsEx**-learnable. Suppose, by way of contradiction, $\mathcal{S}_{\text{SD}}$ is $\tau(\mathbf{Cons})\mathbf{GEx}$-learnable, as witnessed by some $h \in \mathcal{R}$. By **KRT** there is e such that, for all x,

$$\varphi_e(x) = \begin{cases} e, & \text{if } x = 0; \\ 0, & \text{else if } h(\varphi_e[x] \diamond 0) \neq h(\varphi_e[x]); \\ 1, & \text{otherwise.} \end{cases}$$

[5] For notational purposes, we write the argument σ as a subscript to g.

Let $g = \varphi_e$. As h is total, g is total; thus, $g \in \mathcal{S}_{\text{SD}}$. If, for some x, $h(g[x] \diamond 0) = h(g[x])$, then we have

$$\varphi_{h(g[x])}(x) = 0 \neq 1 = g(x).$$

Thus, h on g is infinitely often incorrect (if the last case of the definition of φ_e holds infinitely often) or h on g changes infinitely often its mind, a contradiction.

Regarding "**GConsEx** $\subseteq$ **ItEx**", let $\mathcal{S}$ be **GConsEx**-learnable as witnessed by $h \in \mathcal{P}$. We define an iterative learner h' on previous conjecture e and new datum n, y such that

$$h'(e, n, y) = h(\varphi_e[n] \diamond y).$$

Note that, for all $g \in \mathcal{S}$, we have that the conjectures of h terminate on all inputs less than the length of the data; thus, h' on any such g will make exactly the same sequence of conjectures as h on g. This finishes the proof for "$\subseteq$".

Regarding "**GConsEx** $\neq$ **ItEx**", let $\mathcal{S}$ be such that

$$\mathcal{S} = \{g \in \mathcal{R} \mid \exists e \forall^\infty n : \pi_1(g(n)) = e \wedge \varphi_e = g\}.$$

Recall that this is $\mathcal{S}$ from Theorem 4.7, and that $\mathcal{S}$ is dense. Clearly, $\mathcal{S} \in$ **ItEx**. We now show $\mathcal{S} \notin$ **GConsEx**. Suppose, by way of contradiction, otherwise, as witnessed by $h' \in \mathcal{P}$. As $\mathcal{S}$ is dense, we have $h' \in \mathcal{R}$; furthermore, also from denseness, h' is consistent on all inputs, and, thus, will change its mind on at least one of two inconsistent extensions of any sequence.

By **KRT**, there is e such that, for all x,

$$\varphi_e(x) = \mu y \in \{\langle e, 0\rangle, \langle e, 1\rangle\} . h(\varphi_e[x] \diamond y) \neq h(\varphi_e[x]).$$

As h has to change its mind on at least one of two extensions, we get that φ_e total; thus, $\varphi_e \in \mathcal{S}$. But of course h' does not **GConsEx**-identify φ_e, a contradiction.

Regarding "**ItEx** $\subset$ **GEx**", let $\mathcal{S}$ be

$$\mathcal{S}_{\text{FinSup}} \cup \{g \in \mathcal{R} \mid \varphi_{g(0)} = g \wedge 0 \notin \text{range}(g)\}.$$

This is **GEx**-learnable as follows; a learner would conjecture $\sigma(0)$ for all σ until a 0 appears. From then on, it switches to a learner for the set of all functions of finite support.

Suppose now $\mathcal{S} \in$ **ItEx** as witnessed by some $h \in \mathcal{P}$. Fix a program for h. For each σ, we denote with $h^*(\sigma)$ the output of h after being fed the elements of σ one after the other. Note that $\mathcal{S}$ is dense, as it contains the set of functions of finite support. Thus, h^* is total.

By **KRT**, there is $e \neq 0$ and $(\sigma_i)_{i \in \mathbb{N}}$ such that, for all x,

$$\begin{aligned} \sigma_0 &= e; \\ \forall i : \sigma_{i+1} &= \sigma_i \diamond \mu \tau \in \mathbb{N}_+^* . h^*(\sigma_i \diamond \tau) \neq h^*(\sigma_i); \\ \varphi_e(x) &= \bigcup_{i \in \mathbb{N}} \sigma_i. \end{aligned}$$

Suppose first φ_e is total. Then $\varphi_e \in \mathcal{S}$, but h does not **ItEx**-learn φ_e, a contradiction.

Suppose now φ_e is not total. Then there is i such that σ_i is defined, but σ_{i+1} is not. Then $h^*(\sigma_i \diamond 1) = h^*(\sigma_i \diamond 2)$. As both $\sigma_i \diamond 1 \diamond 0^\infty$ and $\sigma_i \diamond 2 \diamond 0^\infty$ are in $\mathcal{S}$, but h cannot distinguish between the two and converges (if at all) to the same program on both, h fails to identify one of the two, a contradiction. □

Theorem 6.2. We have

(1) **ItEx** $\not\subseteq$ **GConfEx**; and

(2) **GConfEx** $\not\subseteq$ **ItEx**.

The proof of this theorem is left as an exercise.

7 Reliability and Prudence

A restriction for **Ex**-style learning is given by *reliable* learning. Intuitively, a learner is reliable if it never converges to an incorrect conjecture, i.e., always changes its mind after incorrect conjectures. Formally, this is modeled as follows.

$$\textbf{Rel} = \{(p, g) \mid \forall n : \varphi_{p(n)} \neq g \Rightarrow \exists m > n : p(m) \neq p(n)\}.$$

A further restriction on learning is *prudence*. Let I be a learning criterion. A learner h is said to prudently I-learn a set $\mathcal{S}$ iff there is a set $\mathcal{S}' \supseteq \mathcal{S}$ such that

(1) h I-learns all of $\mathcal{S}'$; and

(2) for all $g \in \mathcal{S}'$ and p the I-learning sequence of h on g and all n, $\varphi_{p(n)} \in \mathcal{S}'$.

Intuitively, a prudent learner learns such that every hypothesis is a potential target.

For any learning criterion I we let **Prud**(I) be the set of sets of functions which can be prudently I-learned.

Theorem 7.1. We have

$$\tau(\textbf{Cons})\textbf{GEx} \subset \tau(\textbf{Rel})\textbf{GEx} \subset \textbf{GEx}.$$

Proof. The inclusion "τ(**Cons**)**GEx** $\subseteq$ τ(**Rel**)**GEx**" is trivial. To show the separation, consider the set

$$\mathcal{S} = \{g \in \mathcal{R} \mid \varphi_{g(0)} = g \wedge \forall n : \Phi_{g(0)}(n) \leq g(n+1)\}.$$

With **KRT** we see that $\mathcal{S} \neq \emptyset$. This set can be learnt reliably as follows. Let h be such that, for all $\sigma \neq \emptyset$,

$$h(\sigma) = \begin{cases} \sigma(0), & \text{if } \forall x < \text{len}(\sigma) - 1 : \varphi_{\sigma(0)}\downarrow_{\sigma(x+1)}(x) = \sigma(x); \\ \text{len}(\sigma), & \text{otherwise.} \end{cases}$$

Now suppose, by way of contradiction, that $\mathcal{S} \in \tau$(**Cons**)**GEx**, as witnessed by some learner h. By **KRT**, there is e such that, for all x,

$$\varphi_e(x) = \begin{cases} e, & \text{if } x = 0; \\ \mu y > \Phi_e(x-1) . h(\varphi_e[x] \diamond y) \neq h(\varphi_e[x]), & \text{otherwise.} \end{cases}$$

Using induction, we can now show that φ_e is total, as h needs to change any inconsistent conjecture (so h can only not change its conjecture on a single extension of any sequence). This gives that h cannot learn $\varphi_e \in \mathcal{S}$, a contradiction.

The inclusion "$\tau(\mathbf{Rel})\mathbf{GEx} \subseteq \mathbf{GEx}$" is trivial. To show the separation, we consider $\mathcal{S}_{\mathrm{SD}}$. Suppose, by way of contradiction, $\mathcal{S}_{\mathrm{SD}} \in \tau(\mathbf{Rel})\mathbf{GEx}$, as witnessed by some function h. Note that, for any sequence σ, there is a computable function g extending σ which is not equal to $\varphi_{h(\sigma)}$. Thus, as h is reliable, on g at some time after seeing σ, h will change its mind. With other words,

$$\forall \sigma \exists \tau : h(\sigma \diamond \tau) \neq h(\sigma). \tag{7}$$

Using **KRT**, we get e and $(\sigma_i)_{i\in\mathbb{N}}$ such that, for all x,

$$\begin{aligned} \sigma_0 &= e; \\ \forall i : \sigma_{i+1} &= \sigma_i \diamond \mu\tau.h(\sigma_i \diamond \tau) \neq h(\sigma_i); \\ \varphi_e(x) &= \bigcup_{i\in\mathbb{N}} \sigma_i. \end{aligned}$$

Clearly, from (7), we get φ_e is total; thus, $\varphi_e \in \mathcal{S}_{\mathrm{SD}}$. This is a contradiction, as h does not **RelGEx**-learn φ_e. □

We note at the side that every learner outputting only total conjectures can be made consistent.

Remark 7.2. We have

$$\begin{aligned} \tau(\mathbf{T})\mathbf{GEx} &= \tau(\mathbf{TCons})\mathbf{GEx}; \\ \mathbf{GTEx} &= \mathbf{GTConsEx}. \end{aligned}$$

The remark is easily seen patching.

Theorem 7.3. We have

$$\begin{aligned} \mathcal{T}\mathbf{Prud}(\mathbf{GEx}) &= \tau(\mathbf{T})\mathbf{GEx}; \\ \mathbf{Prud}(\mathbf{GEx}) &= \mathbf{GTEx}. \end{aligned}$$

Proof. The first inclusion comes from the observation that one can learn all subsets of uniformly computable sets of functions prudently (and Theorem 4.7), and that any learner prudent on all input always outputs conjectures for total functions.

Similarly, we get "$\subseteq$" for the second equality.

Let now $\mathcal{S} \in \mathbf{GTEx}$ as witnessed by $h \in \mathcal{R}$. Using Remark 7.2, we suppose that h is consistent on $\mathcal{S}$. Without loss of generality, $\varphi_{h(\emptyset)} \in \mathcal{S}$. We define a function f such that, for all σ,

$$f(\sigma) = \mu t.\varphi_{h(\sigma[t])}[\mathrm{len}(\sigma)] = \sigma.$$

From h consistent on input from $\mathcal{S}$, we get that f is defined on input from $\mathcal{S}$. We define $h' \in \mathcal{R}$ such that, for all σ,

$$h'(\sigma) = h(\sigma[f(\sigma)]).$$

With other words, h' searches through all conjectures that h would have made on the given input, and outputs the first consistent hypothesis.

Let $\mathcal{S}' = \mathcal{S} \cup \{\varphi_{h'(g[t])} \mid g \in \mathcal{S}\}$. We show that $\mathcal{S}'$ witnesses that h' prudently learns $\mathcal{S}$. Let $g \in \mathcal{S}'$. In the case that $g \in \mathcal{S}$, we have that there is a minimal t_0 such that $h(g[t_0])$ is a program for g. Let $t_1 \geq t_0$ be such that, for all $t' < t_0$, the conjecture $h(g[t'])$ is not consistent with $g[t_1]$ (this has to exist, as $g \in \mathcal{S}$ and h is a learner outputting only programs for total functions on input from $\mathcal{S}$, and as t_0 was chosen as the first correct conjecture). Thus, for all $t' > t_1$, $h'(g[t']) = h(g[t_0])$. This shows that h **GEx**-learns $\mathcal{S}$. The set $\{\varphi_{h'(g[t])} \mid g \in \mathcal{S}\}$ is **GEx**-learned by the definition of h'.

Let $g \in \mathcal{S}'$ and let $t \in \mathbb{N}$. We need to show that $h'(g[t])$ is a program for a function in $\mathcal{S}'$. This is trivial for $g \in \mathcal{S}$. Suppose now $g \notin \mathcal{S}$. By the definition of $\mathcal{S}'$, there is $g' \in \mathcal{S}$ such that $g = \varphi_{h'(g'[t])}$. From the definition of h' we see that h' is consistent (on elements from $\mathcal{S}$), so we get $g'[t] = g[t]$. This gives $g = \varphi_{h'(g[t])}$, which shows that $g \in \mathcal{S}$. □

8 Learning by Enumeration

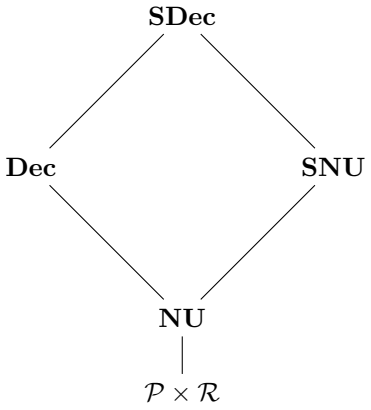
In this section, we first introduce four variants of *decisive learning*, where a learner is forbidden to return to past conjectures, including two notions of *non-U-shaped learning*, where the learner is merely forbidden to return to previous *correct* conjectures. In both cases (not returning to any previous conjecture and not returning to previous correct conjectures) we distinguish two variants depending on what it means for a learner to "leave" or "discard" a conjecture: conjectures can be discarded syntactically (by outputting a syntactically different conjecture, a mind change) or semantically (by outputting a semantically different conjecture). We call the first variation "strong".

Formally, we make the definitions for decisive, strongly decisive, non-U-shaped, and strongly non-U-shaped learning as given in Figure 1. The intuition behind these definitions is as follows. For decisive learning, any conjecture "sandwiched" between two equivalent conjectures has to be equivalent (or even equal) to these conjectures; this implies any conjectures ever discarded is never returned to. For non-U-shaped learning, this is restricted to "sandwiching" between correct conjectures.

The relation between these four sequence acceptance criteria (and the no-restriction, $\mathcal{P} \times \mathcal{R}$) is given by the following diagram.

$$\begin{aligned}
\mathbf{SDec} &= \{(p,g) \mid \forall \ell \leq m \leq n : \varphi_{p(\ell)} = \varphi_{p(n)} \Rightarrow p(\ell) = p(m)\};\\
\mathbf{Dec} &= \{(p,g) \mid \forall \ell \leq m \leq n : \varphi_{p(\ell)} = \varphi_{p(n)} \Rightarrow \varphi_{p(\ell)} = \varphi_{p(m)}\};\\
\mathbf{SNU} &= \{(p,g) \mid \forall \ell \leq m \leq n : \varphi_{p(\ell)} = g = \varphi_{p(n)} \Rightarrow p(\ell) = p(m)\};\\
\mathbf{NU} &= \{(p,g) \mid \forall \ell \leq m \leq n : \varphi_{p(\ell)} = g = \varphi_{p(n)} \Rightarrow \varphi_{p(\ell)} = \varphi_{p(m)}\}.
\end{aligned}$$

Figure 1: Formal definition of the sequence acceptance criteria for strongly decisive, decisive, strongly non-U-shaped, and non-U-shaped learning.



For **GEx**-learning, we get the very strong result that every learner can be assumed strongly decisive, on all possible input (see Theorem 8.5). However, getting this result is far from trivial: we need to have an intimate knowledge about the conjectures we make in order to know when to discard them, and how not to get back to them.

An easy case is a uniformly computable set of functions $\mathcal{S}$. Suppose $p \in \mathcal{R}$ is a list of programs for all and only the functions in $\mathcal{S}$. On input σ, we can just output $p(i)$ for the minimal i such that $\varphi_{p(i)}$ is consistent with σ. Formally, we can define a computable *acceptability principle* A (a computable binary predicate) such that

$$A(i,\sigma) \Leftrightarrow \sigma \subseteq \varphi_{p(i)}. \tag{8}$$

Then A is computable predicate (in dependence on p). The learner $h \in \mathcal{P}$ such that

$$\forall \sigma : h(\sigma) = p(\mu i . A(i,\sigma)) \tag{9}$$

will then successfully learn all functions for which programs are listed by p. The learner h is said to *learn by enumeration*. But what properties of A does h really need to learn $\mathcal{S}$? The following items are sufficient, for a given computable predicate A and a computable p:

(1) For all $g \in \mathcal{S}$ there is an i such that $\varphi_{p(i)} = g$ and $\forall t : A(i, g[t])$.

(2) For all $g \in \mathcal{S}$ and i such that $\varphi_{p(i)} \neq g$, $\forall^\infty t : \neg A(i, g[t])$.

The first item gives the existence of a correct conjecture that always acceptable; the second item gives the (eventual) refutation of all conjectures on inappropriate data. Formally, we call A an *acceptability principle for $\mathcal{S}$ using p* iff (1) and (2) above.

The question is now whether we can find, for any $\mathcal{S} \in \mathbf{GEx}$, an acceptability principle A (using some p). Formally, we say that a learning criterion I *allows for learning by enumeration* iff, for all I-learnable sets $\mathcal{S}$, there exist p and A such that A is an acceptability principle for $\mathcal{S}$ using p. Why does this help us with decisive learning? First, we hope that the acceptability principle is *monotone*, i.e., once it does not accept some conjecture, it will never return to accepting it – this avoids going back to old conjectures; but this we can always assume without loss of generality. Second, we hope that the list p of conjectures is such that, for all i, j, $\varphi_{p(i)} \neq \varphi_{p(j)}$. This will then immediately give that the enumeration learner from (9) is strongly decisive on all input. In particular, such a result makes a strong characterization of **GEx**-learning. The following theorem will lead to such a strong characterization in Corollary 8.3.

Theorem 8.1. Let $\mathcal{S} \in \mathbf{GEx}$. The there is $p \in \mathcal{R}$ and $d \in \mathcal{R}$ such that

(1) for all $g \in \mathcal{S}$ there is i such that $\varphi_{p(i)} = g$; and

(2) for all i, j with $i \neq j$ we have

$$\exists x < d(i,j) : \varphi_{p(i)}(x) \neq \varphi_{p(j)}(x).^6$$

Proof. Let $\mathcal{S} \in \mathbf{GEx}$ as witnessed by $h \in \mathcal{P}$. Without loss of generality, we can assume that h learns an infinite set. Let M be the set of all pairs $\langle z, n\rangle$ such that

- for all $x \leq n$ we have $\varphi_z(x)\downarrow$;
- $h(\varphi_z[n]) = z$; and
- $n = 0$ or $h(\varphi_z[n-1]) \neq h(\varphi_z[n])$.

Note that M is a computably enumerable set. As h learns an infinite set, we have that M is infinite. Thus, there is a

[6] Intuitively, d tells us how far we have to look in order to see that $p(i)$ and $p(j)$ code different functions; in particular, every function is listed at most once.

1-1 enumeration $e \in \mathcal{R}$ of all and only the elements in M. We let $z, n \in \mathcal{R}$ be such that, for all i, $e(i) = \langle z(i), n(i)\rangle$ (i.e., $z = \pi_1 \circ e$ and $n = \pi_2 \circ e$. We define p with s-m-n such that, for all i and x, $\varphi_{p(i)}(x) =$

$$\begin{cases} \varphi_{z(i)}(x), & \text{if } x < n(i) \text{ or, for all } y \text{ with } n(i) < y < x : \\ & \quad \varphi_{z(i)}(y)\downarrow \text{ and } h(\varphi_{z(i)}[y+1]) = z; \\ \uparrow, & \text{otherwise.} \end{cases}$$

We let $d \in \mathcal{R}$ be such that, for all i, j,

$$d(i,j) = \max(n(i), n(j)).$$

We now show that p and d are as desired. Regarding (1), let $g \in \mathcal{S}$. Let t be such that, for all $t' > t$, $h(g[t']) = h(g[t])$; and $t = 0$ or $h(g[t]) \neq h(g[t-1])$. This exists as h **GEx**-learns g; with $s = h(g[t])$ we also get $\varphi_s = g$. Thus, there is i such that $e(i) = \langle s, t\rangle$. From the definition of p we now get $\varphi_{p(i)} = g$.

Regarding (2), let $i, j \in \mathbb{N}$. Without loss of generality, suppose $n(i) \leq n(j)$. Suppose that, for all $x < d(i,j) = n(j)$, we have $\varphi_{p(i)}(x) = \varphi_{p(j)}(x)$; we will show that $i = j$. From the definition of $p(j)$ we get that, for all $x < n(j)$, $\varphi_{p(j)}(x)\downarrow$. In particular, $\varphi_{p(i)}(n(j))\downarrow$ which implies

$$h(\varphi_{z(i)}[n(j)]) = z(i).$$

From $\langle z(j), n(j)\rangle \in M$ we get

$$h(\varphi_{z(j)}[n(j)]) = z(j).$$

These two equations together give $z(i) = z(j)$. Furthermore, $n(i) < n(j)$ would imply that $h(\varphi_{z(j)}[n(j)-1]) = z(j)$, a contradiction to $\langle z(j), n(j)\rangle \in M$, which then gives $n(i) = n(j)$ and, from e being 1-1, $i = j$. □

Theorem 8.2. For all $\mathcal{S} \in$ **GEx** there are p and A such that $\lambda i.\varphi_{p(i)}$ is 1-1 and A is an acceptability principle for $\mathcal{S}$ using p.

Proof. Let $\mathcal{S} \in$ **GEx**, and let p and d as in Theorem 8.1. Define A as follows.

$$A(i,\sigma) \Leftrightarrow \neg(\exists j \leq \text{len}(\sigma) : i \neq j \ \wedge \ d(i,j) \leq \text{len}(\sigma) \wedge \varphi_{p(j)}[d(i,j)]\downarrow_{\text{len}(\sigma)} = \sigma[d(i,j)]).$$

Note that A is computable. We show that A is an acceptability principle for $\mathcal{S}$ using p. Let $g \in \mathcal{S}$. We get that there is i such that $\varphi_{p(i)} = g$ from (1) in Theorem 8.1. Suppose now there is t such that $\neg(A(i, g[t]))$, that is, there is $j \neq i$ such that $d(i,j) \leq t$ and $\varphi_{p(j)}[d(i,j)] = g[d(i,j)]$. This implies $\varphi_{p(j)}[d(i,j)] = \varphi_{p(i)}[d(i,j)]$, a contradiction to the choice of d. Thus, we see

$$\forall t : A(i, g[t]),$$

which shows the first requirement for A to be an acceptability principle.

Regarding the second requirement, let $g \in \mathcal{S}$ and i be such that $\varphi_{p(i)} \neq g$ (in particular, $i \neq j$). Let j be such that $\varphi_{p(j)} = g$ and let t be large enough such that

- $j \leq t$;
- $d(i,j) \leq t$; and
- $\varphi_{p(j)}[d(i,j)]\downarrow_t$.

From this we get, for all $t' \geq t$, $\neg A(i, g[t'])$ as desired. □

The previous two theorems together give us the following characterization of **GEx**-learnability.

Corollary 8.3. A set $\mathcal{S} \subseteq \mathcal{R}$ is **GEx**-learnable if and only if The there is $p \in \mathcal{R}$ and $d \in \mathcal{R}$ such that

(1) for all $g \in \mathcal{S}$ there is i such that $\varphi_{p(i)} = g$; and

(2) for all i, j with $i \neq j$ we have

$$\exists x < d(i,j) : (\varphi_{p(i)}(x) \neq \varphi_{p(j)}(x)).$$

From Theorem 8.2 we also get the following corollary.

Corollary 8.4. **GEx** allows for learning by enumeration.

Theorem 8.5. We have

$$\tau(\textbf{SDec})\textbf{GEx} = \textbf{GEx}.$$

Proof. We use the enumeration learner given by Theorem 8.2. □

Another popular learning criterion is conservative learning; for conservative learning, a learner is never allowed to discard a conjecture while there is no *direct* evidence that the conjecture is incorrect, i.e., that the conjecture is inconsistent. We make the following formal definition.

$$\textbf{Conv} = \{(p,g) \mid p \text{ tot. } \wedge \forall i : p(i+1) \neq p(i) \Rightarrow g[i] \not\subseteq \varphi_{p(i)}\}.$$

As a corollary to the proof of Theorem 8.2, we get the following.

Corollary 8.6. We have

$$\textbf{GConvEx} = \textbf{GEx}.$$

Proof. In the proof of Theorem 8.2 we constructed an acceptability predicate, which will make sure that no consistent hypotheses are discarded. □

We can also give a characterization, similar to that of Corollary 8.3, for **GConsEx**-learning.

Theorem 8.7. A set $\mathcal{S} \subseteq \mathcal{R}$ is **GConsEx**-learnable if and only if The there is $p \in \mathcal{R}$ such that

(1) for all $g \in \mathcal{S}$ there is i such that $\varphi_{p(i)} = g$; and

(2) for all i, σ such that σ is extensible to a function from $\mathcal{S}$, we have that $\sigma \subseteq \varphi_{p(i)}$ is decidable.

Proof. The direction "$\Leftarrow$" follows by enumeration with the acceptability principle from (8).

Let $\mathcal{S} \in \mathbf{GConsEx}$ as witnessed by h. We let p be as in Theorem 8.1. Then we have, for all σ, i, that $\sigma \subseteq \varphi_{p(i)}$ iff, for all $x < \text{len}(\sigma)$,

$$(x < n(i) \ \wedge \ \varphi_{z(i)}(x) = \sigma(x)) \ \vee \ h(\sigma[x+1]) = z(i).$$

This is computable as, for all $x < n(i)$, $\varphi_{z(i)}(x)\downarrow$; it is equivalent to $\sigma \subseteq \varphi_{p(i)}$ as h is consistent (on relevant σ). □

9 Other Criteria

The literature contains a wealth of other learning criteria. We will give two more here. First, one can require a **GEx**-learner to converge not to just *any* description of the input language, but a *minimal* one, in accordance with the principle of Occam's Razor (stating that one should always go for the simplest explanation of the available data). As it turns out, requiring learning of the actually minimal correct program is then dependent on what hypothesis space is used (i.e., what machine model is used). With the following definition, one can bypass this phenomenon. For all $r \in \mathcal{R}$, let

$$\mathbf{Mex}_r = \{(p,g) \mid \exists e \leq r(\text{minProg}(g)) : \forall^\infty n : p(n) = e \ \wedge \ \varphi_e = g\}.^7$$

Intuitively, $\mathbf{Mex}_r$ allows for a hypothesis "somewhat" larger than the minimal one, where "somewhat" is defined by r. However, even when allowing very quickly growing functions r, we cannot achieve the full learning power of **GEx**, i.e.,

$$\bigcup_{r \in \mathcal{R}} \mathbf{GMex}_r \subset \mathbf{GEx}.$$

A learning criterion that tries to find a middle ground between converging to a single correct conjecture (as in **Ex**) or converging to "infinitely many" correct conjectures (as in **Bc**). This learning criterion, named **Fex** allows for oscillation between a *finite set* of correct conjectures. Formally, we let

$$\mathbf{Fex} = \{(p,g) \mid \exists D \text{ finite} : \forall^\infty n : p(n) \in D \wedge \forall e \in D : \varphi_e = g\}.$$

Regarding **GFex**-learning, we get the following result.

Theorem 9.1. We have

$$\mathbf{GEx} = \mathbf{GFex}.$$

Proof. The inclusion "$\subseteq$" is trivial. Regarding "$\supseteq$", let $\mathcal{S} \in \mathbf{GFex}$ as witnessed by h. For all σ, we define $D \in \mathcal{R}$ such that

$$D(\sigma) = \{p \mid \exists x < \text{len}(\sigma) : p = h(\sigma[x]) \wedge \neg(\exists y < \text{len}(\sigma) : \varphi_p(x)\downarrow_{\text{len}(\sigma)} \neq \sigma(y))\}.$$

That is, $D(\sigma)$ lists all the conjectures made by h so far, for which it has not found an error, a direct inconsistency so far. Note that, for all $g \in \mathcal{S}$, there is t large enough such that $D(g[t])$ will contain at least one correct index for g, all indices do not contradict g, and for larger t, the set D will not change any more.

We let $m \in \mathcal{R}$ be a function such that, for all finite sets F and all x, if there is $e \in F$ such that $\varphi_e(x)\downarrow$, then $\varphi_{m(F)}(x) = \varphi_e(x)$ for the first such e found.

From the remarks regarding D it is now clear that $\lambda\sigma.m(D(\sigma))$ will **GEx**-learn $\mathcal{S}$. □

10 Language Learning

In this section we will make a quick exploration of language learning. A target to be learned is in this setting a *formal language*, a computably enumerable subset of $\mathbb{N}$. As hypotheses we use natural numbers, where each number e represents the set

$$W_e = \text{dom}(\varphi_e).$$

It is well known that $(W_e)_{e \in \mathbb{N}}$ is an enumeration of all and only the computably enumerable sets. There is an effective procedure such that, for each e, the procedure on input e lists all and only the elements of W_e, without repetitions.

Learning criteria for language learning are almost the same as for function learning, with some important differences. Learners are still all elements from $\mathcal{P}$; learnees are computably enumerable $L \subseteq \mathbb{N}$. A learner cannot take in an infinite language all at once, so it has to be presented piece by piece, just as with function learning. However, for function learning, there was a canonical order in which to present the data; for language learning, this is a bit more difficult. We will consider learning from *texts*, but there are other possibilities. A *text* for a language L is an infinite listing of all and only the elements of L, plus, possibly, the *pause symbol* #. The pause symbol signals that no new data is available. This makes the infinite sequence of pause symbols the only text for the empty language (all other languages have pause-free texts).

A learner h successfully learns a language L iff it successfully learns L from *any* text for L (including non-computable). Learner restrictions and sequence generating operators work just as in function learning (with the texts as target functions). The sequence acceptance criteria are a bit different, basically substituting W for φ and any text T is replaced by content(T), the content of

T (all non-# elements listed by T). For example, **Ex** for language learning is defined as

$$\mathbf{Ex} = \{(p,T) \mid \exists e \forall^\infty n : p(n) = e \ \wedge \ W_e = \text{content}(T)\}.$$

Another example is consistent learning as follows.

$$\mathbf{Cons} = \{(p,T) \mid \forall n : \text{content}(T[n]) \subseteq W_{p(n)}\}.$$

As a first example, note that the set of all finite languages is **TxtGEx**-learnable. However, for any infinite language L, the set of L and all finite subsets of L is *not* **TxtGBc**-learnable.

However, there are some sequence generating operators for language learning that are not applicable to function learning. We will introduce set-driven learning and partially set-driven learning below. The idea is that order in which the data is presented should not matter to the learner, but only the set of elements of presented data sequence (for set-driven learning) or additionally the length of the sequence of data (for partially set-driven learning). For a learner h, a text T and all n, we let

$$\begin{aligned} \mathbf{Sd}(h,T)(n) &= h(\text{content}(T[n])); \\ \mathbf{Psd}(h,T)(n) &= h(\text{content}(T[n]),n). \end{aligned}$$

As the first theorem about language learning, we show that set-driven learning is weaker than learning from sequences.

Theorem 10.1. We have

$$\mathbf{TxtSdEx} \subset \mathbf{TxtGEx}.$$

Proof. The inclusion is trivial; we show the separation as follows. Let $\mathcal{L}$ consist of all languages $\{\langle e,x\rangle \mid x \in \mathbb{N}\}$ where $\varphi_e(0)\uparrow$ as well as all the finite sets $\{\langle e,x\rangle \mid x \leq \varphi_e(0)\}$ where $\varphi_e(0)\downarrow$. This is **TxtGEx**-learnable by checking for longer and longer time whether $\varphi_e(0)\downarrow$ (outputting an index for $\{\langle e,x\rangle \mid x \in \mathbb{N}\}$ while this did not converge), and, if convergent, just output and index for the finite set of data seen.

Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{TxtSdEx}$ as witnessed by $h \in \mathcal{P}$. By KRT, there is e such that $\varphi_e(0)$ equals the first m found such that h on $\{\langle e,x\rangle \mid x \leq m\}$ gives and output e such that W_e includes an element outside of $\{\langle e,x\rangle \mid x \leq m\}$. With other words, we look at whether h overgeneralizes, and as soon as it does, we make it be incorrect.

In the first case, $\varphi_e(0)\uparrow$, in which case h on the text $\langle e,0\rangle, \langle e,1\rangle \ldots$ for $\{\langle e,x\rangle \mid x \in \mathbb{N}\}$ will never make a conjecture coding an infinte set, so it cannot correctly identify $\{\langle e,x\rangle \mid x \in \mathbb{N}\} \in \mathcal{L}$.

In the second case, $\varphi_e(0)\downarrow$, in which case $\{\langle e,x\rangle \mid x \leq \varphi_e(0)\} \in \mathcal{L}$, but h on this (finite) set makes a hypothesis including incorrect data, which shows that h does not learn it.

This shows that $\mathcal{L}$ is not **TxtSdEx**-learnable. □

The proof that we have just seen used computational difficulties of the learner: if the halting problem was decidable, then the set would have been learnable. We now get to a different kind of problem for language learning which is studied using the notion of a *locking sequence*. Let L be a language and h a learner. Then we say that σ is a *locking sequence for h on L* iff $\text{content}(\sigma) \subseteq L$, $W_h(\sigma) = L$ and, for all τ with $\text{content}(\tau) \subseteq L$, $h(\sigma \diamond \tau) = h(\sigma)$.

As a major result on language learning we get the following theorem, sometimes called the *Locking Lemma*. Its proof is basically an application of *König's Infinity Lemma*.

Theorem 10.2. Let $h \in \mathcal{P}$, $L \in \mathbf{TxtGEx}(h)$ and σ a sequence with $\text{content}(\sigma) \subseteq L$. Then there is σ' extending σ such that σ' is a locking sequence for h on L.

Proof. Suppose, by way of contradiction, otherwise. Then we can keep extending σ to a text T for L such that h on T makes infinitely many mind changes, a contradiction to $L \in \mathbf{TxtGEx}(h)$. □

A learner h is called *order-independent* iff, for all languages L such that there exists a text for L on which h converges to some e, then, for *all* texts T for L, h on T converges to e.

Theorem 10.3. Let $\mathcal{L}$ be **TxtGEx**-learnable. Then there is a order-independent h such that $\mathcal{L} \subseteq \mathbf{TxtPsdEx}(h)$.

Proof. Let $h_0 \in \mathcal{P}$ be a **TxtGEx**-learner for $\mathcal{L}$. Let P be a computable predicate such that, for all σ, D, t, $P(\sigma,D,t)$ is true iff, for all σ' extending σ with $\text{content}(\sigma') \subseteq D$ and $\text{len}(\sigma') < t$, $h_0(\sigma) = h_0(\sigma')$. In a sense, σ' *appears to be* a locking sequence. For given D and t, we let $f(D,t)$ be the least σ with $\text{content}(\sigma) \subseteq D$ such that $P(\sigma,D,t)$. Note that such a σ always exists, as long enough σ's do not have any extensions of length $< t$.

We now let $h \in \mathcal{P}$ be such that, for all D,t, $h(D,t) = h_0(f(D,t))$. It is now easy to see that, on each text T for a language L, f on T converges to the least locking sequence σ_0 of h on any text of L. Thus, on any text T for L, h on T converges to $h_0(\sigma_0)$. □

References

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

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

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

[JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Massachusetts, second edition, 1999.

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

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967. Reprinted by MIT Press, Cambridge, Massachusetts, 1987.

[Sho01] J.R. Shoenfield. *Recursion Theory*. Lecture notes in logic. A.K. Peters, 2001.