On characterizations of learnability with computable learners

Tom F. Sterkenburg   TOM.STERKENBURG@LMU.DE
Munich Center for Mathematical Philosophy
Ludwig Maximilian University of Munich

Editors: Po-Ling Loh and Maxim Raginsky

Abstract

We study computable PAC (CPAC) learning as introduced by Agarwal et al. (2020). First, we consider the main open question of finding characterizations of proper and improper CPAC learning. We give a characterization of a closely related notion of *strong* CPAC learning, and provide a negative answer to the COLT open problem posed by Agarwal et al. (2021) whether all decidably representable VC classes are improperly CPAC learnable. Second, we consider undecidability of (computable) PAC learnability. We give a simple general argument to exhibit such undecidability, and initiate a study of the arithmetical complexity of learnability. We briefly discuss the relation to the undecidability result of Ben-David et al. (2019), that motivated the work of Agarwal et al.

Keywords: PAC learning, VC dimension, computability, undecidability, arithmetical complexity

1. Introduction

What changes in the theoretical analysis of learning algorithms when we impose a restriction to algorithms that are, in fact, algorithmic? This fundamental question led Agarwal et al. (2020) to initiate a study of statistical learning theory with computable learners. The theory of probably approximately correct (PAC) learning, as presented by Shalev-Shwartz and Ben-David (2014), is founded on the Vapnik-Chervonenkis (VC) theory of uniform convergence (1971), that separates the statistical analysis of learning functions from computational considerations. On the other hand, PAC learning draws its name from Valiant's computational approach (1984; see Kearns and Vazirani, 1994), that focuses on the efficiency (polynomial runtime) of learners. Agarwal et al. introduce a natural intermediate set-up, where it is (only) required for learners to be computable functions. They obtain several results about the ensuing notion of computable PAC (CPAC) learning and its relationship to unconstrained PAC learnability.

The fundamental theorem of PAC learning (Blumer et al., 1989) states that (under mild measurability conditions) a class of hypotheses is PAC learnable precisely if it satisfies the combinatorial property of finite VC dimension. Moreover, a class is PAC learnable precisely if the procedure of empirical risk minimization (ERM) PAC learns it. The main lesson that Agarwal et al. draw from their results is that the computability requirement "disrupts the fundamental characterization of learnability by the finite VC-dimension of a class" (2020, p. 59). However, they leave as an open question what conditions *do* characterize computable PAC learnability. As the two most important questions for future research, they ask for characterizations of proper and of improper CPAC learnability. The latter motivates the open problem announced by Agarwal et al. (2021), whether there are decidably representable PAC learnable classes that are not even improperly CPAC learnable.

In the first main part of this paper (Section 3), we make progress on these two questions. We introduce a notion of *strong* CPAC (SCPAC) learnability, by adding a stipulation on the computability

of the sample complexity. The motivation for this notion is that we can prove a natural characterization (that does preserve the classical characterization as neatly as possible), namely as the conjunction of finiteness of VC dimension and computability of ERM. In fact, the notions of CPAC and SCPAC learnability are so close that they may already be equivalent; we leave this as an open question. Further, we solve the open problem of Agarwal et al. (2021). We confirm their conjecture that a particular decidably representable PAC learnable class is not even improperly CPAC learnable, implying that there is a nontrivial question of characterizing improper CPAC learnability.

An incentive for the work of Agarwal et al. was the result due to Ben-David et al. (2017, 2019) that learnability can be undecidable. Ben-David et al. introduce a general learning model of "estimating the maximum" (EMX), and exhibit a particular EMX learnability problem that they prove to be independent of the ZFC axioms of set theory (provided ZFC is consistent). From this result they infer that "there is no VC dimension-like parameter that generally characterizes learnability" (2019, p. 44). Their analysis is that "the source of the problem is in defining learnability as the existence of a learning function rather than the existence of a learning algorithm" (ibid., p. 48). In the same vein, Agarwal et al. (2020, p. 48) write that "[h]ad we required learners to be computable, there would have been a finite representation for each learner [...], ruling out independence of ZFC results of the type shown in Ben-David et al. (2017, 2019)."

In the second main part of this paper (Section 4), we turn to the undecidability of computable PAC learnability. On the basis of Rice's Theorem, we offer a simple argument to the effect that, for *any* notion of learnability in the current computable framework, and a general approach to formulating decision problems of learnability (computable families of hypothesis classes), the resulting decision problem, if not trivial (either every class is learnable or every class is not), is unsolvable. We observe that the unsolvability of a learnability decision problem directly entails that the learnability of infinitely many hypothesis classes is independent of the ZFC axioms (provided ZFC is arithmetically sound). Further, we initiate an investigation (similar to the work of Beros, 2014; Beros et al., 2021 for algorithmic learning theory) into *how* undecidable learnability problems are: that is, into their arithmetical complexity. In particular, we use our characterization of SCPAC learnability to show that this decision problem is Σ_3-complete. Finally (in Section 5), we briefly discuss how our observations relate to the undecidability result of Ben-David et al.

Related work   We restrict attention to the framework of Agarwal et al. (2020), where the domain set is countable and hypotheses are total computable functions (see Section 2). Ackerman et al. (2021) present results about computable PAC learning within a more general framework of computable analysis, where the domain is an arbitrary computable metric space. They also remark on the assumption of a computable sample complexity, the added ingredient in our notion of SCPAC learning. Calvert (2015) already studied a computable setting where the domain is 2^ω and hypotheses are Π_1^0 classes, and established the arithmetical complexity of PAC learnability (finiteness of VC dimension) of effective hypothesis classes within this setting. Calvert further notes the relation to earlier work on the computational complexity of calculating the VC dimension of finite hypothesis classes over finite domain (Linial et al., 1991; Schaefer, 1999). Schaefer, citing Wehner (1990), also gives the arithmetical complexity of PAC learnability within the computable setting we study here. Caro (2021) recently showed the undecidability of (among other models) PAC learning, constructing instances of both "Turing undecidability" (unsolvability of decision problem) and "Gödel undecidability" (independence of axiom system). His constructions for the undecidability of PAC learning apply to the current computable setting, and indeed the relevant (families of) hypothesis

classes are computable, but they only partly transfer to CPAC learnability (see Section 4.1 for more details). Beros (2014); Beros et al. (2021) study the arithmetical complexity of learnability for the algorithmic learning theory paradigm of identification in the limit (Gold, 1967; Jain et al., 1999).

2. Preliminaries

2.1. PAC learning

Let $\mathcal{X} = \mathbb{N}$ the domain, and $\mathcal{Y} = \{0, 1\}$ the label space. A hypothesis is a function $h : \mathcal{X} \to \mathcal{Y}$. A *sample S* is a finite ordered sequence of input-label pairs, or formally, $S \in \mathcal{S} := \cup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$. To assess hypotheses, we use the 0/1 error function. Thus the *error* of h on sample S is given by

$$L_S(h) := \frac{|\{(x, y) \in S : h(x) \neq y\}|}{|S|},$$

and the true error or *risk* of h w.r.t. a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ is $L_\mathcal{D}(h) := \mathbb{P}_{(x,y) \sim \mathcal{D}}[h(x) \neq y]$.

Definition 1 (PAC learnability) *A hypothesis class $\mathcal{H}$ is PAC learnable if there exists a function $m_\mathcal{H} : (0, 1)^2 \to \mathbb{N}$ and a learning function $A : \mathcal{S} \to \mathcal{H}$ such that for all $\epsilon, \delta \in (0, 1)$, for all $m \geq m_\mathcal{H}(\epsilon, \delta)$ and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ we have*

$$\text{Prob}_{S \sim \mathcal{D}^m} \left[L_\mathcal{D}(A(S)) \leq \min_{h \in \mathcal{H}}(L_\mathcal{D}(h)) + \epsilon \right] \geq 1 - \delta. \tag{1}$$

We also call the above *agnostic* PAC learning to distinguish it from the more specific case of *realizable* PAC learning, where we make the assumption that there exists $h^* \in \mathcal{H}$ with $L_\mathcal{D}(h^*) = 0$. We also call the above *proper* PAC learning to distinguish it from the more general case of *improper* PAC learning, where we do not assume that the range of the learning function A is restricted to $\mathcal{H}$. That is, A may also output hypotheses that are not in $\mathcal{H}$; but condition (1), including the comparison to the best hypothesis *in $\mathcal{H}$*, does not change.

Definition 2 *Empirical risk minimization for hypothesis class $\mathcal{H}$, write $\text{ERM}_\mathcal{H}$, returns for each $S \in \mathcal{S}$ a hypothesis in $\arg\min_{h \in \mathcal{H}} L_S(h)$.*

For hypothesis class $\mathcal{H}$ and $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}$, the *restriction of $\mathcal{H}$ to X* is the class $\mathcal{H}_{|X}$ of functions $f : X \to \mathcal{Y}$ such that $f(x) = h(x)$ for some $h \in \mathcal{H}$ and all $x \in \mathcal{X}$. We say that $\mathcal{H}$ *shatters* finite $X \subset \mathcal{X}$ if the restriction of $\mathcal{H}$ to X contains *all* functions $f : X \to \mathcal{Y}$.

Definition 3 *The VC dimension of hypothesis class $\mathcal{H}$, write VCdim($\mathcal{H}$), is the maximal size of a set $X \subset \mathcal{X}$ that is shattered by $\mathcal{H}$. If $\mathcal{H}$ shatters sets of arbitrarily large size, then VCdim($\mathcal{H}$) = ∞.*

Theorem 4 (Fundamental theorem of PAC learning, Blumer et al., 1989) *A hypothesis class $\mathcal{H}$ is PAC learnable if and only if $\text{ERM}_\mathcal{H}$ PAC learns $\mathcal{H}$ if and only if VCdim($\mathcal{H}$) < ∞.*

2.2. Computable PAC learning

We use the following computability-theoretic notation (see, e.g., Soare, 2016). Let $\{\phi_i\}_{i \in \mathbb{N}}$ be a standard enumeration of all partial computable (p.c.) functions. We write $\phi_i(x) \downarrow = y$ to denote that ϕ_i halts on input x and returns y, while $\phi_i(x) \uparrow$ denotes that ϕ_i does not halt on x. We write

$\phi_{i,s}(x) = y$ if ϕ_i outputs y on input x within s computation steps; by convention, $i, x, y < s$. We similarly write $\phi_{i,s}(x) \downarrow$ if ϕ_i has halted and produced an output on x by s or $\phi_{i,s}(x) \uparrow$ if it has not.

In computable PAC (CPAC) learning, we work with *computable* hypotheses, total computable functions $h : \mathcal{X} \to \mathcal{Y}$. Moreover, learners must be actual learning *algorithms*, total computable functions from samples to computable hypotheses.

Definition 5 (CPAC learnability, Agarwal et al., 2020) *A hypothesis class $\mathcal{H}$ is CPAC learnable if there exists a* total computable $A : \mathcal{S} \to \mathcal{H}$ *that PAC learns $\mathcal{H}$.*

We again also use the terms *agnostic* and *proper* to distinguish this notion from the more specific case of realizable CPAC learning and the more general case of improper CPAC learning.

The following fact is an immediate consequence of Theorem 4 and Definition 5.

Fact 1 *If $VCdim(\mathcal{H}) < \infty$ and $\mathrm{ERM}_{\mathcal{H}}$ is computably implementable, i.e., there is a total computable function that computes a version of $\mathrm{ERM}_{\mathcal{H}}$, then $\mathcal{H}$ is CPAC learnable.*

We further introduce a variant of CPAC learning, that we call *strong* CPAC (or SCPAC) learning, where it is explicitly stipulated that the learning algorithm comes with a computable sample complexity function. We discuss the motivation for this notion in Section 3.1.

Definition 6 (SCPAC learnability) *A hypothesis class $\mathcal{H}$ is SCPAC learnable if there exists a* total computable $A : \mathcal{S} \to \mathcal{H}$ *and a total computable $m_{\mathcal{H}} : \mathbb{N}^2 \to \mathbb{N}$ such that for all $a, b \in \mathbb{N}$, for all $m \geq m_{\mathcal{H}}(a, b)$ and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$,*

$$\mathrm{Prob}_{S \sim \mathcal{D}^m} \left[L_{\mathcal{D}}(A(S)) > \min_{h \in \mathcal{H}}(L_{\mathcal{D}}(h)) + 1/a \right] < 1/b. \tag{2}$$

The sufficient condition of Fact 1 is already sufficient for SCPAC learnability.

Proposition 7 *If $VCdim(\mathcal{H}) < \infty$ and $\mathrm{ERM}_{\mathcal{H}}$ is computably implementable, then $\mathcal{H}$ is SCPAC learnable.*

Proof Given $\mathcal{H}$ with $VCdim(\mathcal{H}) = d < \infty$, Sauer's lemma gives us a computable bound (depending only on the finite information d) on the sample complexity for the uniform convergence property of $\mathcal{H}$ (Shalev-Shwartz and Ben-David, 2014, Theorem 6.7), which in turn gives us a computable bound on the sample complexity of $\mathrm{ERM}_{\mathcal{H}}$ (ibid., Corollary 4.4). ∎

2.3. Computability of hypothesis classes

We would also like to formulate a notion of effective computability of hypothesis classes, classes of computable hypotheses. Namely, a class of total computable functions can *itself* fail to be computable, in the sense that there is no computable way of checking or even enumerating its elements.

Example 1 (Agarwal et al., 2020, Theorem 9) *Define for $i \in \mathbb{N}$ hypothesis h_i by*

$$h_i(x) = \begin{cases} 1 & \text{if } x = 2i \text{ or } x = 2i + 1 \ \& \ \phi_i(i) \downarrow \\ 0 & \text{otherwise} \end{cases}$$

and let hypothesis class $\mathcal{H}_{halt} := \{h_i\}_{i \in \mathbb{N}}$*. While each* individual h_i *is computable (since given by finite information), the* h_i *are not* uniformly *computable in* i *(or we could solve the Halting problem), meaning the members of* $\mathcal{H}_{halt}$ *cannot be computably enumerated. This underlies the fact that* $\mathcal{H}_{halt}$ *is not CPAC learnable, not even in the realizable case. Namely, by Fact 1 it would suffice for CPAC learnability that* $\mathrm{ERM}_{\mathcal{H}_{halt}}$ *is computably implementable. For this, in the realizable case, it would suffice that the elements of* $\mathcal{H}_{halt}$ *can be enumerated (Agarwal et al., 2020, Theorem 10).*

As a general approach to a notion of effective hypothesis classes, we always assume some encoding that computably corresponds the natural numbers (indices) to programs (Turing machines) for computing hypotheses, inducing some *base class* $\hat{\mathcal{H}}$ of computable hypotheses. More precisely, we assume a computable *decoding function* $C : \mathbb{N} \to \hat{\mathcal{H}}$, that gives a computable listing $\{h_i\}_{i \in \mathbb{N}} = \hat{\mathcal{H}}$ by $h_i := C(i)$. Note that this base class $\hat{\mathcal{H}}$ must always be a strict subclass of the class $\mathcal{H}_{\mathrm{comp}}$ of *all* computable hypotheses, because the computable hypotheses are the total computable (t.c.) functions, and by a standard diagonalization argument we cannot effectively enumerate or encode (programs for) all and only the t.c. functions.

Given such an encoding of a base class $\hat{\mathcal{H}}$, the available hypothesis classes $\mathcal{H} \subseteq \hat{\mathcal{H}}$ correspond to (and can be identified with) the subsets of $\mathbb{N}$. A computable subset of $\mathbb{N}$ then gives a computable class of (codes of) hypotheses in $\hat{\mathcal{H}}$, and a c.e. subset of $\mathbb{N}$ gives a computable enumeration of (codes of) a class of hypotheses in $\hat{\mathcal{H}}$. The former corresponds to the notion of *decidably representable* (DR) hypothesis class of Agarwal et al. (2020), and the latter to their notion of *recursively enumerably representable* (RER) hypothesis class. We will adopt this terminology.

Example 2 *Consider the base class* $\mathcal{H}_{\mathrm{fin}}$ *of all hypotheses* with finite support*: the hypotheses* h *with* x_0 *such that* $h(x) = h(x')$ *for all* $x, x' > x_0$ *(Agarwal et al., 2020, Remark 7). Each such* h *is given by the finite information of its corresponding* x_0*, the list of labels for* $x \le x_0$*, and the constant label for* $x > x_0$*; and we can clearly specify an encoding of all such hypotheses that gives a computable decoding function* $C : \mathbb{N} \to \mathcal{H}_{\mathrm{fin}}$*. Examples of DR subclasses—or choices of base classes in their own right—are the class* $\mathcal{H}_{\mathrm{ivl}}$ *of* interval hypotheses *(h with* x_0, x_1 *such that* $h(x) = 1$ *iff* $x_0 < x < x_1$*) and the class* $\mathcal{H}_{\mathrm{thd}}$ *of* threshold hypotheses *(h* $\in \mathcal{H}_{\mathrm{ivl}}$ *with* $x_0 = 0$*). An example of a non-DR RER subclass is the class of threshold hypotheses with* x_1 *such that* $\phi_{x_1}(x_1) \downarrow$*.*

Not every DR hypothesis class is CPAC learnable (Agarwal et al., 2020, Theorem 11), which means there are DR classes for which ERM is not computably implementable. On the other hand, a hypothesis class does not have to be RER to be SCPAC learnable.

Example 3 *Take the class* $\mathcal{H}_{\mathrm{ivl}}$ *of interval hypotheses. This class has VC dimension 2 and* $\mathrm{ERM}_{\mathcal{H}_{\mathrm{ivl}}}$ *is clearly computably implementable, so it is SCPAC learnable. Now extend this class with all threshold functions* h_a *such that* $\phi_a(a) \uparrow$*. The extended class* $\mathcal{H}'$ *is no longer RER. However,* $\mathrm{VCdim}(\mathcal{H}_{\mathrm{ivl}}) = \mathrm{VCdim}(\mathcal{H}')$ *and we have that for each* $S \in \mathcal{S}$*,* $\min_{h \in \mathcal{H}_{\mathrm{ivl}}} L_S(h) = \min_{h \in \mathcal{H}'} L_S(h)$*, so that the algorithm for* $\mathrm{ERM}_{\mathcal{H}_{\mathrm{ivl}}}$ *also implements* $\mathrm{ERM}_{\mathcal{H}'}$*. Thus* $\mathcal{H}'$ *is also SCPAC learnable.*

3. Towards characterizations of computable learnability

3.1. Proper (S)CPAC learnability

We saw that a hypothesis class is (S)CPAC learnable if it has finite VC dimension and ERM is computably implementable. For SCPAC learnability, this condition pair is also necessary.

STERKENBURG

Theorem 8 *A hypothesis class $\mathcal{H}$ is properly SCPAC learnable if and only if VCdim$(\mathcal{H}) < \infty$ and there exists an algorithm that implements* ERM$_{\mathcal{H}}$.

Proof It remains to show the left-to-right direction. So suppose $\mathcal{H}$ is SCPAC learnable. Then $\mathcal{H}$ is PAC learnable, so VCdim$(\mathcal{H}) < \infty$; and there are computable learning function A and computable sample complexity function $m_{\mathcal{H}}$ such that for all $a, b \in \mathbb{N}$, for all $m \geq m_{\mathcal{H}}(a, b)$ and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ we have (2). Using A, we can computably implement ERM$_{\mathcal{H}}$ as follows.

For given training sample $S = ((x_1, y_1), \ldots, (x_n, y_n))$, define distribution $\mathcal{D}_S$ by $\mathcal{D}_S((x_i, y_i)) = 1/n$ for each $(x_i, y_i) \in S$ (in case of repetitions in S, we simply add up the probabilities). Choose $a > n$ and any b, and compute $m = m_{\mathcal{H}}(a, b)$. Let $\mathcal{S}_{\mathcal{D}_S}^m$ be the set of all possible length-m samples that can be generated from $\mathcal{D}_S$. By running A on all these sequences, we can computably pick some $\hat{S} \in \arg\min_{S' \in \mathcal{S}_{\mathcal{D}_S}^m} L_S(A(S'))$. The claim is that $\hat{h} = A(\hat{S})$ is also in $\arg\min_{h \in \mathcal{H}} L_S(h)$. Namely, if not, then for all $S' \in \mathcal{S}_{\mathcal{D}_S}^m$ we would have $L_S(A(S')) > \min_{h \in \mathcal{H}} L_S(h)$. Specifically, each $A(S')$ would make at least one more mistake on S than the $h \in \arg\min_{h \in \mathcal{H}} L_S(h)$, which by definition of $\mathcal{D}_S$ implies $L_{\mathcal{D}_S}(A(S')) \geq \min_{h \in \mathcal{H}} L_{\mathcal{D}_S}(h) + 1/n$. But that implies that with certainty ($\mathcal{D}_S^m$-probability 1) we would sample $S' \sim \mathcal{D}_S$ of length m with $L_{\mathcal{D}_S}(A(S')) > \min_{h \in \mathcal{H}} L_{\mathcal{D}_S}(h) + 1/a$, contradicting (2). ∎

We do not know whether CPAC learnability is not already equivalent to SCPAC learnability. If it is not, then Theorem 8, which constitutes an effective version of the original equivalence between PAC learnability and PAC learnability by ERM, gives reason for thinking that SCPAC learnability is a natural notion. Moreover, the above proof suggests that an $\mathcal{H}$ that is CPAC but not SCPAC learnable has extreme properties. In particular, it can only be learnable by an algorithm A for which we cannot compute an upper bound on any corresponding sample complexity function $g_b(a) = m(a, b)$ for fixed b. That is to say, the sample complexity must grow faster in a than any computable function.

Question 1 *Does there exist a hypothesis class that is properly CPAC learnable but not properly SCPAC learnable?*

In any case, both the negative and the positive results on CPAC learning in (Agarwal et al., 2020) also go through for SCPAC learning: the first (Theorems 9 and 11) because the latter is stronger, the second (Theorems 10, 13, and 15; Corollary 14) because they rely on showing the computable implementability of ERM, which already gives SCPAC learnability.

3.2. Improper (S)CPAC learnability

We now turn to the improper case. To be clear, we use the qualifier "improper" as a generalization of "proper." We will use the qualifier "strictly improper" to mean "improper but not proper." The following fact is immediate from the definitions.

Fact 2 *If $\mathcal{H}$ is improperly CPAC learnable and $\mathcal{H}' \subseteq \mathcal{H}$ then $\mathcal{H}'$ is improperly CPAC learnable. The same holds for improper SCPAC learnability.*

Agarwal et al. (2020, Theorem 9, Theorem 11) exhibit two classes $\mathcal{H}_{\text{halting}}$ and $\mathcal{H}_{\text{LT}}$ that are not properly CPAC learnable, yet that *are* improperly (so *strictly* improperly) CPAC (indeed SCPAC) learnable (ibid., p. 59). Intuitively, the reason is that the incomputable information encoded in these classes can be "blotted out" by adding more hypotheses. This is easy if (as in the case of $\mathcal{H}_{\text{halting}}$ and $\mathcal{H}_{\text{LT}}$) a class only contains, for some constant b, hypotheses (seen as sets of positively labeled

instances) of size bounded by b. Then the obvious SCPAC learnability of the superclass of *all* such b-bounded-size hypotheses means by Fact 2 that the original class is improperly SCPAC learnable.[1]

In general, by Fact 2, extendability to a proper (S)CPAC learnable class is sufficient for improper (S)CPAC learnability; the next question, towards an actual characterization, is whether it is actually a *necessary* condition (Agarwal et al., 2020, Conjecture 23). But a preceding question is whether, at least for RER hypothesis classes, there is not already a more trivial characterization: *every* RER class with finite VC dimension is improperly (S)CPAC learnable. We show here that this is not the case. For this purpose we take the hypothesis class $\mathcal{H}_{\text{init}}$ defined by Agarwal et al. (2021, p. 4641), which they already conjecture is not even improperly CPAC learnable (ibid., Conjecture 9). We slightly reformulate their definition. Let, for each $s \in \mathbb{N}$, computable hypothesis h_s be defined by

$$h_s(x) = \begin{cases} 1 & \text{if } \phi_{x,s}(x) \downarrow \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathcal{H}_{\text{init}} := \{h_s\}_{s \in \mathbb{N}}$. This class is in fact DR and has VC dimension 1. First we need a lemma.

Lemma 9 *If $\mathcal{H}$ is improperly CPAC learnable, then for sufficiently large n, we can computably find for any $X = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ of size n a function $g : \{x_1, \ldots, x_n\} \to \{0, 1\}$ with $g \notin \mathcal{H}_{|X}$.*

Proof Suppose there exists an algorithm A that improperly learns $\mathcal{H}$. Picking some $a > 8$ and $b > 7$, that means that there is sufficiently large $m = m(\epsilon, \delta)$ such that for any $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$

$$\text{Prob}_{S \sim \mathcal{D}^m} \left[L_{\mathcal{D}}(A(S)) \geq \min_{h \in \mathcal{H}}(L_{\mathcal{D}}(h)) + 1/8 \right] < 1/7.$$

But by the computable No-Free-Lunch Theorem (Agarwal et al., 2020, Lemma 19), for any $X = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ of size $n = 2m$ we can computably find a function $g : \{x_1, \ldots, x_n\} \to \{0, 1\}$ such that for distribution $\hat{\mathcal{D}}$ uniform over $\{(x_1, g(x_1)), \ldots, (x_n, g(x_n))\}$ we have

$$\text{Prob}_{S \sim \hat{\mathcal{D}}^m} \left[L_{\hat{\mathcal{D}}}(A(S)) \geq 1/8 \right] \geq 1/7.$$

This implies that $g \notin \mathcal{H}_{|X}$, for else $\min_{h \in \mathcal{H}}(L_{\hat{\mathcal{D}}}(h)) = 0$ and we would have a contradiction. ∎

Theorem 10 *The class $\mathcal{H}_{\text{init}}$ is not improperly CPAC learnable.*

Proof Suppose it is. Then by Lemma 9 there exists, for some sufficiently large n of our choice, an algorithm B that for any n input elements $x_1, \ldots, x_n$ proceeds as follows. If $x_i \neq x_j$ for all distinct $i, j \leq n$, then B returns a function $g : \{x_1, \ldots, x_n\} \to \{0, 1\}$ such that $g \notin \mathcal{H}_{\text{init}|\{x_1, \ldots, x_n\}}$. If not, then B returns some default function on $\{x_1, \ldots, x_n\}$, say the constant-0 function.

We define, for each $i \leq n$, a total computable n-place f_i such that

$$\phi_{f_i(x_1, \ldots, x_n)}(z) = \begin{cases} i & \text{if } z = 0 \\ 0 & \text{if } z = x_i > 0 \ \& \ B(x_1, \ldots, x_n)(x_i) = 1 \\ \uparrow & \text{otherwise.} \end{cases}$$

---

1. A class need not have this boundedness property for similar reasoning to go through, as shown by an example of one of the referees. For any b-bounded-size $\mathcal{H} = \{h_i\}_i$ define $\mathcal{H}' = \{h'_i\}_i$ by $h'_i(x) = h_i(x)/2$ if x is even and $h'_i(x) = 1$ otherwise, yielding a class of infinite hypotheses that is nevertheless extendable to a properly CPAC learnable class. An interesting further question is to find a more concrete characterization of such extendability.

Now by the n-fold Recursion Theorem (Smullyan, 1993, p. 117) there are $c_1, \ldots, c_n > 0$ such that for each $i \leq n$,

$$\phi_{c_i} = \phi_{f_i(c_1, \ldots, c_n)}.$$

Moreover, these $c_1, \ldots, c_n$ must be distinct, else $\phi_{f_i(c_1, \ldots, c_n)} = \phi_{f_j(c_1, \ldots, c_n)}$ for some $i \neq j$, which is excluded by the fact that they have distinct range (for each i only $\phi_{f_i(c_1, \ldots, c_n)}$ has i in its range). But then for function $g = B(\{c_1, \ldots, c_n\})$ we have for each c_i that $\phi_{c_i}(c_i) \downarrow$ iff $\phi_{f_i(c_1, \ldots, c_n)}(c_i) \downarrow$ iff $g(c_i) = 1$. This means there exists a large enough s such that for each $i \leq n$, $\phi_{c_i, s}(c_i)$ iff $g(c_i) = 1$, which implies by definition of $\mathcal{H}_{\text{init}}$ that $g \in \mathcal{H}_{\text{init}|\{c_1, \ldots, c_n\}}$, contrary to specification of B. ∎

It follows with Fact 2 that there are RER classes with finite VC dimension that cannot be extended to properly (S)CPAC learnable classes. So the latter extendability property is in this sense nontrivial; the question remains whether it actually characterizes improper (S)CPAC learnability.

Question 2 (Agarwal et al., 2020, Conjecture 23) *Does there exist a (RER) class $\mathcal{H}$ that is not extendable to a properly (S)CPAC learnable class, yet that is improperly (S)CPAC learnable?*

4. Undecidability and complexity of learnability

4.1. Undecidability

There are two kinds of undecidability, that are related but not the same (see, e.g., Poonen, 2014, p. 211; Hamkins, 2020, pp. 251–52; Caro, 2021).

1. *Independence of a statement from an axiom system.* A statement Y is independent of (or undecidable in) axiom system $\mathcal{A}$ if neither Y nor its negation can be derived from these axioms using the rules of logic. That is, neither $\mathcal{A} \vdash Y$ nor $\mathcal{A} \vdash \neg Y$. An example is the independence of the continuum hypothesis from the ZFC axioms of set theory.

2. *Unsolvability of a decision problem.* A decision problem, i.e., a family $\{Q_i\}_{i \in \mathbb{N}}$ of problems with YES/NO answers, is unsolvable (or undecidable) if there is no decision algorithm that on each input $i \in \mathbb{N}$ returns the correct answer to Q_i. The standard example is the unsolvability of the Halting problem, that asks for each $i \in \mathbb{N}$ whether $\phi_i(i) \downarrow$.

Let "learnable" in this section stand for *any* specific notion of learnability. We first consider the undecidability of learnability in sense (2), or the unsolvability of a *learnability decision problem*.

To a first approximation, a learnability decision question asks: does there exist a decision algorithm that for every given hypothesis class returns YES if the class is learnable and returns NO if it is not? To make this question meaningful at all, we must presuppose some family H of hypothesis classes such that each $\mathcal{H} \in \mathsf{H}$ can actually be presented as input to a candidate decision algorithm.

Example 4 *It is impossible to effectively encode the family H^{all} of all hypothesis classes of computable hypotheses. A learnability problem for H^{all} is therefore trivially undecidable: there exists no decision algorithm, because there cannot even exist an algorithm to query on each $\mathcal{H} \in \mathsf{H}^{\text{all}}$.*

8

Let a *computable* family $\mathsf{H} = \{\mathcal{H}_j\}_{j \in \mathbb{N}}$ of hypothesis classes be such that there is a computable procedure that for each given $j \in \mathbb{N}$ retrieves an effective representation of $\mathcal{H}_j$; at the least, it uniformly retrieves an instruction for enumerating the elements of $\mathcal{H}_j$ (so the hypothesis classes of a computable family are all RER). For any computable family $\{\mathcal{H}_j\}_{j \in \mathbb{N}}$ we can clearly state a corresponding *decidability of learnability question*: does there exists an algorithm that for each input j returns YES if $\mathcal{H}_j$ is learnable and NO otherwise?

We describe a general way of constructing computable families of hypothesis classes, and show that for each family constructed in this way, the decision problem, if not trivial, is undecidable. Pick any base class $\hat{\mathcal{H}} \subset \mathcal{H}_{\mathrm{comp}}$ of computable hypotheses that we can code onto the natural numbers. The uniformly c.e. family $\{W_i\}_{i \in \mathbb{N}}$ of all c.e. subsets of $\mathbb{N}$, or equivalently the family $\{\phi_i\}_{i \in \mathbb{N}}$ of all p.c. functions, picks out the computable family $\mathsf{H} = \{\mathcal{H}_i\}_{i \in \mathbb{N}}$ of all RER hypothesis classes $\mathcal{H}_i \subseteq \hat{\mathcal{H}}$. We call such a family H a *maximal* computable family of hypothesis classes. Importantly, such a maximal family $\mathsf{H} = \{\mathcal{H}_i\}_{i \in \mathbb{N}}$ has the property that if $\phi_i = \phi_j$ then also $\mathcal{H}_i = \mathcal{H}_j$.

Now the answer to our question is *yes*, for any computable family that either only contains learnable or only contains unlearnable hypotheses classes. For such a family that is *trivial for learnability*, either the constant YES algorithm or the constant NO algorithm is a decision algorithm.

Example 5 *The maximal computable family constructed from the base class $\mathcal{H}_{\mathrm{ivl}}$ of interval hypothesis is a trivial family for PAC learnability: already the base class has finite VC dimension. This family is also trivial for improper (S)CPAC learning (as the base class is SCPAC learnable, Example 3). However, the family is nontrivial for proper (S)CPAC learning: there exist RER classes of interval hypotheses that are not CPAC learnable (Agarwal et al., 2020, Theorem 11).*

But as soon as a maximal computable family is nontrivial for learnability, the answer is *no*.

Proposition 11 *For any particular notion of learnability, and any maximal computable family H of hypotheses classes that is nontrivial for this learnability, the learnability problem is unsolvable.*

Proof By the correspondence between the members of H and all p.c. functions, this follows directly from Rice's Theorem (see Soare, 2016, p. 16) that every *nontrivial index set* is incomputable. An index set $I \subseteq \mathbb{N}$ is a set of indices of p.c. functions closed under extensional equivalence,

$$i \in I \ \& \ \phi_i = \phi_j \Longrightarrow j \in I,$$

and nontrivial if neither $I = \emptyset$ nor $I = \mathbb{N}$. Now for any maximal computable family $\mathsf{H} = \{\mathcal{H}_i\}_{i \in \mathbb{N}}$ of hypothesis classes, we have that if $\mathcal{H}_i$ is learnable and $\phi_i = \phi_j$, then $\mathcal{H}_i = \mathcal{H}_j$ and $\mathcal{H}_j$ is learnable, too; so that the set $I_{L(\mathsf{H})} = \{i \in \mathbb{N} : \mathcal{H}_i \text{ learnable}\}$ is an index set, that is non-trivial if H is. But then Rice's Theorem says that $I_{L(\mathsf{H})}$ is incomputable, which just means that there can be no decision algorithm that for every i returns YES if $i \in I_{L(\mathsf{H})}$ and NO otherwise. ■

Undecidability is not limited to maximal computable families as constructed above.

Example 6 (Caro, 2021, Section 2.3) *Caro constructs a computable family $\mathsf{H}_{\mathrm{halt}} = \{\mathcal{H}_{M_j}\}_{j \in \mathbb{N}}$ uniformly from the class $\{M_j\}_{j \in \mathbb{N}}$ of Turing machines (i.e, the class $\{\phi_j\}_{j \in \mathbb{N}}$ of p.c. functions), and proves undecidability of the PAC learnability problem for $\mathsf{H}_{\mathrm{halt}}$. This family also has the property that $\phi_i = \phi_j$ implies $\mathcal{H}_{M_i} = \mathcal{H}_{M_j}$, so that the previous reasoning by Rice's Theorem actually applies here too. Caro's own proof is a direct derivation of the undecidability of finiteness of VC dimension for $\mathsf{H}_{\mathrm{halt}}$, which entails undecidability of PAC learnability and also (as noted by Caro,*

2021, Section 5) of realizable CPAC learnability, as both are characterized by finite VC dimension (for RER classes). In fact, by Theorem 8, finite VC dimension here already characterizes (agnostic) SCPAC learnability, because one can verify that all classes in $\mathsf{H}_{\mathrm{halt}}$ *admit of a computable implementation of ERM. Still, the advantage of the generality of the reasoning by Rice's Theorem is that it directly gives us undecidabililty for* any *learnability notion that* $\mathsf{H}_{\mathrm{halt}}$ *is nontrivial for.*

Caro also already showed undecidability of PAC learning in sense (1).

Example 7 (Caro, 2021, Section 2.2) *Caro presents a construction, for any sufficiently expressive formal system F, of an RER hypothesis class $\mathcal{H}_F$ such that $\mathcal{H}_F$ has finite VC dimension if and only if F is consistent. Since, by Gödel's second incompleteness theorem, F (provided it is consistent) does not decide its own consistency, this yields, for any F, that F does not decide the learnability of $\mathcal{H}_F$. In particular, ZFC (provided it is consistent) does not decide the learnability of $\mathcal{H}_{ZFC}$.*

As Caro (2021, Remark 2.24) also notes, there is a way of directly deriving undecidability in sense (1) from undecidability in sense (2); so in particular from Proposition 11. We follow the reasoning outlined by Poonen (2014, pp. 212–13).

Proposition 12 *Given any particular notion of learnability that we can arithmetically characterize (which includes PAC learnability and SCPAC learnability, see Section 4.2). For any computable family $\mathsf{H} = \{\mathcal{H}_i\}_{i \in \mathbb{N}}$ of hypothesis classes such that the learnability decision problem is unsolvable (in particular, any maximal computable family for which this learnability is nontrivial), the learnability of infinitely many $\mathcal{H}_i$ is independent of ZFC (provided ZFC is arithmetically sound).*

Proof Using the presupposed characterization of the relevant notion of learnability, we can write a computable procedure that for each i returns a statement Y_i of first-order arithmetic that expresses that $\mathcal{H}_i$ is learnable. (For instance, for PAC learnability, the algorithm produces the statement (3) in Section 4.2 below, uniformly plugging in arithmetical representations of the relevant "atomic" statements about computable objects, like $[h(x) \neq y_i]$.) If ZFC is arithmetically sound, it only proves such statements (suitably recast in the language of set theory) that are in fact true. Thus we have a computable procedure that for each i returns a statement Y_i such that

- if $\mathrm{ZFC} \vdash Y_i$ then $\mathcal{H}_i$ is learnable;

- if $\mathrm{ZFC} \vdash \neg Y_i$ then $\mathcal{H}_i$ is not learnable.

But this gives us a decision procedure for learnability for H (for each i enumerate theorems of ZFC until we find either Y_i or $\neg Y_i$), *unless* some (indeed infinitely many) Y_i are independent of ZFC. ∎

4.2. Arithmetical complexity

The general proof by Rice's Theorem of undecidability of learnability does not use any specific properties of the notion(s) of learnability. The mathematical structure of learnability does come into play when we ask the natural next question, namely *how* undecidable learnability is. Specifically, what is the *arithmetical complexity* of the relevant index set (see Soare, 2016)?

We start with standard PAC learnability, characterized by finiteness of VC dimension. We can spell out the property $\mathrm{VCdim}(\mathcal{H}) < d$ as

$$(\forall \text{ distinct } x_1, \ldots, x_d \in \mathcal{X})(\exists y_1, \ldots, y_d \in \{0, 1\})(\forall h \in \mathcal{H})(\exists i \leq d)\, [h(x_i) \neq y_i]. \qquad (3)$$

Since only the first and the third quantifiers are unbounded, this is equivalent to a Π_1 statement. Then the property $\text{VCdim}(\mathcal{H}) < \infty$, equivalent to $(\exists d)[\text{VCdim}(\mathcal{H}) < d]$, is a Σ_2 property. This gives an upper bound on the arithmetical complexity for any computable family of hypothesis classes.

Fact 3 *The problem of PAC learnability for a computable family of hypothesis classes is no harder than Σ_2.*

Moreover, this bound is strict: as observed before by Schaefer (1999) there are computable families of hypothesis classes such that the problem is Σ_2-complete. The following proof is similar to that of Schaefer (1999, theorem 4.1) with reference to Wehner (1990), and is also implicit in Zhao (2018).

Proposition 13 (Schaefer, 1999) *There exists a computable family of hypothesis classes such that the problem of PAC learnability is Σ_2-complete.*

Proof We exhibit a computable family $\mathsf{H} = \{\mathcal{H}_i\}_j$ for which the index set $\{j \in \mathbb{N} : \mathcal{H}_j \text{ is learnable}\}$ is equal to the index set $\text{Fin} = \{j \in \mathbb{N} : W_j \text{ is finite}\}$. The latter is well-known to be Σ_2-complete (see Soare, 2016, p. 86).

Let $\mathcal{H}_{\text{fin}} = \{h_i\}_{i \in \mathbb{N}}$ a computable enumeration of all hypotheses with finite support and $\{W_j\}_{j \in \mathbb{N}}$ an enumeration of all c.e. sets. For every $j \in \mathbb{N}$ define c.e.

$$N_j := \{n \in \mathbb{N} : n \leq |W_j|\} = \{n \in \mathbb{N} : (\exists s)[n \leq |W_{j,s}|]\},$$

and let $\mathcal{H}_j := \{h_i : i \in N_j\}$. Then we have that $j \in \text{Fin}$ precisely if $\text{VCdim}(\mathcal{H}_j) < \infty$. Namely, if $j \in \text{Fin}$ then also $|N_j| = |\mathcal{H}| < \infty$ and $\text{VCdim}(\mathcal{H}_j) < \infty$. But if $j \notin \text{Fin}$ then $|N_j| = \mathbb{N}$ and $\mathcal{H}_j = \mathcal{H}_{\text{fin}}$, so $\text{VCdim}(\mathcal{H}_j) = \infty$. ∎

Next, we turn to SCPAC learnability. Recall its characterization, Theorem 8, by the conjunction of finiteness of VC dimension and the computable implementability of ERM. We first introduce as a lemma an equivalent statement of the second conjunct, that we can then express arithmetically to give us an upper bound.

Lemma 14 *For computable hypothesis class $\mathcal{H}$, $\text{ERM}_\mathcal{H}$ is computably implementable if and only if $B_\mathcal{H} := \{S \in \mathcal{S} : (\exists h \in \mathcal{H})\,[L_S(h) = 0]\}$ is computable.*

Proof We have that $S \in B_\mathcal{H}$ precisely if $L_S(\text{ERM}_\mathcal{H}(S)) = 0$, so it is immediate that if $\text{ERM}_\mathcal{H}$ is computable, then so is $B_\mathcal{H}$. Conversely, if the latter is computable, then the following procedure gives an algorithm for $\text{ERM}_\mathcal{H}$. For given $S = (x^n, y^n)$, for all $i \leq n$, $j \leq \binom{n}{i}$ define $z_{i,j}^n$ to be the j-th length-n binary sequence that disagrees with y^n on precisely i positions. Now for the $i \leq n$ in increasing order, check for each defined $z_{i,j}^n$ whether $(x^n, z_{i,j}^n) \in B_\mathcal{H}$; as soon as this is the case for some $z_{i,j}^n$, start enumerating hypotheses in $\mathcal{H}$ until finding an h with $L_S(h) = i$, and return this h. This procedure will always halt and return a hypothesis $\hat{h} \in \min_{h \in \mathcal{H}} L_S(h)$. ∎

Proposition 15 *The problem of SCPAC learnability for a computable family of hypothesis classes is no harder than Σ_3.*

Proof Let $\langle \cdot \rangle : \mathcal{S} \to \mathbb{N}$ be some computable 1-1 encoding of all finite samples onto the natural numbers. Given computable family $\mathsf{H} = \{\mathcal{H}_j\}_j$, c.e. subset $B_i := \{\langle S \rangle : S \in \mathcal{S} \ \& \ (\exists h \in \mathcal{H}_i)\,[L_S(h) = 0]\} \subseteq \mathbb{N}$ is computable precisely if $B_{\mathcal{H}_i}$ is. Since (cf. Soare, 2016, p. 83)

$$
\begin{aligned}
(\exists d)\,[\overline{W_d} = B_i] &\iff (\exists d)[B_i \cap W_d = \emptyset \wedge B_i \cap W_d = \mathbb{N}] \\
&\iff (\exists d)\,[(\forall s)[B_{i,s} \cup W_{d,s} = \emptyset] \wedge (\forall x)(\exists s)[x \in B_{i,s} \cap W_{d,s}]] \\
&\iff (\exists)\,[(\forall)[\dots] \wedge (\forall)(\exists)[\dots]] \\
&\iff (\exists)(\forall)(\exists)[\dots],
\end{aligned}
$$

the computability of B_i can be expressed as as a Σ_3 statement. But then the conjunction with the Σ_2 statement of finiteness of VC dimension is also a Σ_3 statement. ∎

Again, this bound is strict.

Proposition 16 *There exists a computable family of hypothesis classes such that the problem of SCPAC learnability is Σ_3-complete.*

Proof We show for a family $\{\mathcal{H}_j\}_{j \in \mathbb{N}}$ of classes of threshold functions that the question of SCPAC learnability is equivalent to the index set $\mathrm{Rec} = \{j \in \mathbb{N} : W_j \text{ is computable}\}$, which is Σ_3-complete (Rogers, 1967, Theorem XVI; also see Soare, 2016, p. 88). Recall that, for $i \in \mathbb{N}$, threshold function h_i is defined by by $h(x) = 1$ if and only if $x < i$. In addition, let h_ω be such that $h_\omega(x) = 1$ for all x. From the standard enumeration $\{W_j\}_{j \in \mathbb{N}}$ of the c.e. sets, define $\mathcal{H}_j := \{h_i : i \in W_j\} \cup \{h_\omega\}$.

Since each $\mathcal{H}_j$ has finite VC dimension, SCPAC learnability of $\mathcal{H}_j$ is equivalent to the computability of $B_{\mathcal{H}_j}$. Moreover, $B_{\mathcal{H}_j}$ is computable precisely if $\mathcal{H}_j$ is. Namely, starting with the right-to-left direction, to decide $h_i \in \mathcal{H}_j$ for $i \in \mathbb{N}$ (for h_ω the answer is always yes), it is enough to ask whether $((i,1),(i+1,0)) \in B_{\mathcal{H}_j}$. Conversely, to decide $S \in B_{\mathcal{H}_j}$, we can distinguish four cases. First, if $y = 1$ for all $(x,y) \in S$, then $S \in B_{\mathcal{H}_j}$ because $L_S(h_\omega) = 0$. Second, if there are $(x,0),(x',1) \in S$ with $x < x'$ then $S \notin B_{\mathcal{H}_j}$. Third, if $y = 0$ for all $(x,y) \in S$ then take the smallest x_0 with $(x,y) \in S$; now $S \in B_{\mathcal{H}_j}$ precisely if $h_x \in \mathcal{H}$ for some $x < x_0$. Otherwise, take the $(x_0,y_0),(x_1,y_1) \in S$ with $x_0 < x_1$ and $y_0 = 1, y_1 = 0$ that have smallest difference $|x_0 - x_1|$; now $S \in B_{\mathcal{H}_j}$ precisely if $h_x \in \mathcal{H}$ for some x with $x_0 \leq x < x_1$.

In sum, we have that $j \in \mathrm{Rec}$ iff $\mathcal{H}_j$ is computable iff $\mathcal{H}_j$ is SCPAC learnable. ∎

If Question 1 has a negative answer then the notions of CPAC and SCPAC learnability coincide, and we also have the complexity of the former. Otherwise, we need some different arithmetical characterization for CPAC learning. Similarly, to find the complexity of improper (S)CPAC learnability, we first need an arithmetical characterization of this notion (which would follow from a negative answer to Question 2).

5. Conclusion and discussion

In the first part of this paper, we made progress on the main open problems concerning computable PAC (CPAC) learning: to give characterizations of (im)proper CPAC learnability. We gave a characterization of proper strong CPAC (SCPAC) learning, that is an effective version of the fundamental theorem of PAC learning; and we confirmed the conjecture that there are decidably representable PAC learnable classes that are not even improperly CPAC learnable. We leave as open questions

whether every CPAC learnable class is already SCPAC learnable (in which case we already have a characterization of CPAC learnability) and whether every improperly CPAC learnable class is extendable to a properly CPAC learnable class (in which case we have a characterization of improper CPAC learnability). A natural further question of characterization concerns the notion of nonuniform CPAC learning (Soloveichik, 2008; Agarwal et al., 2020), including a strong variant.

In the second part, we investigated undecidability of (computable) PAC learning. We gave a basic argument to uncover both undecidability of learnability decision problems and the independence of ZFC of learnability, and we initiated a study of the arithmetical complexity of notions of learnability. Future characterizations of notions of learnability (e.g., of improper (S)CPAC learning or nonuniform (S)CPAC learning) also unlock the question of their arithmetic complexity.

What do our observations about undecidability mean for the motivating claim of Agarwal et al. (2020), that the ingredient of computability rules out "independence of ZFC results of the type shown in Ben-David et al. (2017, 2019)"? Proposition 12 does state that for infinitely many particular RER $\mathcal{H}$ the learnability of $\mathcal{H}$ is independent of ZFC (provided ZFC is arithmetically sound). We did not exhibit any *particular* such class, but this is also not hard to do (recall Example 7 of the class $\mathcal{H}_{\mathrm{ZFC}}$ of Caro, 2021). Perhaps the main difference with the original result of Ben-David et al. is that undecidable learnability statements in the computable framework of Agarwal et al. are in the end all statements of first-order arithmetic. Ben-David et al. showed that the EMX learnability of a particular hypothesis class is equivalent to the continuum hypothesis CH—or at least to a weak version of the CH (see Hart, 2019)—which is a more complex set-theoretical statement.

This is important for the conclusion of Ben-David et al. that there is no combinatorial characterization of EMX learning, thus, that there exists "no general dimension for learning" (2019, p. 47). They write that a combinatorial "dimension for learning" (like VC dimension for PAC learning) is a "finite character property" (defined as ZFC-provably equivalent to a bounded formula in the language of set theory, or Δ_0 in the Lévy hierarchy; see Jech, 2003) that does not vary over different models of ZFC (pp. 47–48). On a closer look (2017, p. 14), Ben-David et al. restrict attention to a class of models of ZFC such that Δ_0 properties have the same truth value in each model (these properties are "absolute," in particular, for the class of transitive models of ZFC; see again Jech, 2003). Under this restriction, "loosely speaking, PAC learnability does not depend on the specific model of set theory," whereas "EMX learnability heavily depends on the cardinality of the continuum" and (provided ZFC is consistent) disagreeing models of ZFC "are known to exist" (2017, p. 14–15).

Now Proposition 12 does also directly imply (provided ZFC is arithmetically sound) that for infinitely many particular RER $\mathcal{H}$ there are different models of ZFC that disagree on whether VCdim($\mathcal{H}$) $< \infty$ (whether $\mathcal{H}$ is PAC learnable). However, such disagreeing models, that must involve nonstandard models of arithmetic, are excluded by the above restriction of models. Here we enter the slippery territory of questions of truth and existence in mathematics (some entries to the relevant literature are Koellner, 2009; Button and Walsh, 2018; Hamkins, 2020). Most scholars in the foundations of mathematics would indeed find it implausible to claim that there is no truth to the arithmetical matter of whether a certain RER $\mathcal{H}$ is PAC learnable (has finite VC dimension), just because this is not settled among all (nonstandard) models of arithmetic. Even if we cannot pin it down with first-order axioms, they would argue, we have a clear conception of the natural numbers as per the intended, standard model. Things are much more contentious when it comes to set theory and the continuum hypothesis. While it is therefore *more* plausible to make the analogous claim about the non-existence of a dimension concept for EMX learnability, Ben-David et al. do still commit here to a philosophical position that is hardly uncontroversial.

Acknowledgments

References

Nathanael L. Ackerman, Julian Asilis, Jieqi Di, Cameron E. Freer, and Jean-Baptiste Tristan. On computable learning of continuous features. arXiv: 2111.14630, 2021.

Sushant Agarwal, Nivasini Ananthakrishnan, Shai Ben-David, Tosca Lechner, and Ruth Urner. On learnability wih computable learners. In Aryeh Kontorovich and Gergely Neu, editors, *Proceedings of the 31st International Conference on Algorithmic Learning Theory, ALT 2020*, volume 117 of *Proceedings of Machine Learning Research*, pages 48–60, 2020.

Sushant Agarwal, Nivasini Ananthakrishnan, Shai Ben-David, Tosca Lechner, and Ruth Urner. Open problem: Are all VC-classes CPAC-learnable? In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of the 34rd Conference on Learning Theory, COLT 2021*, volume 134 of *Proceedings of Machine Learning Research*, pages 4636–4641, 2021.

Shai Ben-David, Pavel Hrubeš, Shay Moran, Amir Shpilka, and Amir Yehudayof. A learning problem that is independent of the set theory ZFC axioms. arXiv: 1711.05195, 2017.

Shai Ben-David, Pavel Hrubeš, Shay Moran, Amir Shpilka, and Amir Yehudayoff. Learnability can be undecidable. *Nature Machine Intelligence*, 1:44–48, 2019.

Achilles A. Beros. Learning theory in the arithmetical hierarchy. *The Journal of Symbolic Logic*, 79(3):908–927, 2014.

Achilles A. Beros, Konstantinos A. Beros, Daniel Flores, Umar Gaffar, David J. Webb, and Soowhan Yoon. Learning theory in the arithmetical hierarchy II. *Archive for Mathematical Logic*, 60:301–315, 2021.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36 (4):929–965, 1989.

Tim Button and Sean Walsh. *Philosophy and Model Theory*. Oxford University Press, 2018.

Wesley Calvert. PAC learning, VC dimension, and the arithmetic hierarchy. *Archive for Mathematical Logic*, 54(7-8):871–883, 2015.

Matthias C. Caro. Undecidability of learnability. arXiv: 2106.01382, 2021.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

Joel David Hamkins. *Lectures on the Philosophy of Mathematics*. MIT Press, 2020.

Klaas P. Hart. Machine learning and the continuum hypothesis. *Nieuw Archief voor Wiskunde*, 20 (3):214–217, 2019.

Sanjay Jain, Daniel N. Osherson, James S. Royer, and Arun Sharma. *Systems That Learn: An Introduction to Learning Theory*. A Bradford Book. MIT Press, 2nd edition, 1999.

Thomas Jech. *Set Theory*. Springer Monographs in Mathematics. Springer, third millenium edition, 2003.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

Peter Koellner. Truth in mathematics: The question of pluralism. In Otávio Bueno and Øystein Linnebo, editors, *New Waves in Philosophy of Mathematics*, New Waves in Philosophy, pages 80–116. Palgrave Macmillan, 2009.

Nathan Linial, Yishay Mansour, and Ronald L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. *Information and Computation*, 90(1):33–49, 1991.

Bjorn Poonen. Undecidable problems: a sampler. In Juliette Kennedy, editor, *Interpreting Gödel: Critical Essays*, pages 211–241. Cambridge University Press, 2014.

Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

Marcus Schaefer. Deciding the Vapnik–Červonenkis dimension is Σ_3^p-complete. *Journal of Computer and System Sciences*, 58(1):177–182, 1999.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Raymond M. Smullyan. *Recursion Theory for Metamathematics*, volume 22 of *Oxford Logic Guides*. Oxford University Press, 1993.

Robert I. Soare. *Turing Computability: Theory and Applications*, volume 4 of *Theory and Applications of Computability*. Springer, 2016.

David Soloveichik. Statistical learning of arbitrary computable classifiers. arXiv: 0806.3537, 2008.

Leslie. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

Vladimir N. Vapnik and Alexey Ja. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971. Translation of the Russian original in *Teoriya Veroyatnostei i ee Primeneniya*, 16(2): 264–279, 1971.

S. Wehner. *Zur Komplexität des Numerierens*. PhD Dissertation, Universität Karlsruhe, 1990.

Kino Zhao. A statistical learning theory approach to a problem of induction. PhilSci-Archive: 15256, 2018.