Open Problem: Are all VC-classes CPAC learnable?

Sushant Agarwal SUSHANT.AGARWAL@UWATERLOO.CA
Nivasini Ananthakrishnan NANANTHA@UWATERLOO.CA
Shai Ben-David SHAI@UWATERLOO.CA*
Tosca Lechner TLECHNER@UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, Canada

Ruth Urner RUTH@EECS.YORKU.CA
Lassonde School of Engineering, EECS Department
York University, Toronto, ON, Canada

Editors: Mikhail Belkin and Samory Kpotufe

Abstract

A few years ago, it was shown that there exist basic statistical learning problems whose learnability can not be determined within ZFC (Ben-David et al., 2017, 2019). Such independence, and the implied impossibility of characterizing learnability of a class by any combinatorial parameter, stems from the basic definitions viewing learners as arbitrary functions. That level of generality not only results in unprovability issues but is also problematic from the perspective of modeling practical machine learning, where learners and predictors are computable objects. In light of that, it is natural to consider learnability by *algorithms* that output *computable predictors* (both learners and predictors are then representable as finite objects). A recent study initiated a theory of such models of learning (Agarwal et al., 2020). It proposed the notion of *CPAC learnability*, by adding some basic computability requirements into a PAC learning framework. As a first step towards a characterization of learnability in the CPAC framework, Agarwal et al. (2020) showed that CPAC learnability of a binary hypothesis class is not implied by the finiteness of its VC-dimension anymore, as far as proper learners are concerned. A major remaining open question is whether a similar result holds also for *improper learning*. Namely, does there exist a computable concept class consisting of computable classifiers, that has a finite VC-dimension but no computable learner can PAC learn it (even if the learner is not restricted to output a hypothesis that is a member of the class)?

Keywords: Computability, PAC learning, VC-Dimension

1. Introduction

A few years ago, a study established the existence of basic learning problems whose learnability (even in the sense of *weak learning*) cannot be determined by the common notions of mathematical proofs, namely the set theory ZFC (Ben-David et al., 2017, 2019). A closer look reveals that the standard notion of statistical learnability, initiatiated by Vapnik and Chervonenkis, in which (PAC) learnability is *characterized* by the finiteness of the VC-dimension, refers to *learners* as general functions from training samples to predictors, which also are just functions (Vapnik and Chervonenkis, 1971). Had we required learners to be *computable functions*, there would have been a finite representation for each learner (as the code for the program implementing it), ruling out independence of ZFC results of the type shown in Ben-David et al. (2017, 2019).

* Also, faculty member at the Vector Institute, Toronto

Strengthening the definitions of learnability by allowing only computable learners, arguably would also better reflect our intention of modeling machine learning. Valiant's computational learning theory framework of PAC learnability combined the statistical success condition with a requirement that learners are algorithms whose running time is polynomial in $\frac{1}{\epsilon}, \log\left(\frac{1}{\delta}\right)$ and some parameter d of the function class, for example, the Euclidean dimension of the feature space in the case of linear classifiers (Valiant, 1984; Haussler, 1992). A recent study initiated the investigation of an intermediate setup in the setting of binary classification: *learnability with computable learners* that are *required to output computable predictors*, but are not otherwise restricted by any efficiency requirements (Agarwal et al., 2020). This framework, which we formally review in the next section, was termed *computable PAC (CPAC) learnability*.

Of course, restricting the set of candidate learners from general functions to computable functions can only result in a decrease in the scope of learnable classes. Indeed, it was shown that there exist classes of finite VC-dimension that cannot be learned by any *proper* computable learner in the CPAC setup Agarwal et al. (2020). That is, for proper computable learners (where the learner is required to always output a function from the hypothesis class) VC-dimension is not a sufficient condition for CPAC learnability. However, the question of whether this is also the case for a general learning setup, where the learner is not required to output a function from the fixed hypothesis class, remained open:

Open Question Are there decidable classes $\mathcal{H}$ with finite VC-dimension that are not CPAC learnable in the agnostic case, even when the learner is allowed to output arbitrary computable classifiers?

As shown in Agarwal et al. (2020) major results of the usual unrestricted learning setup break down upon restricting the learners to being computable. It is intriguing to develop a theory of computable learning and figure out which of the other fundamental results in the PAC setup have counterparts in the CPAC model. In particular:

Open Question Is there a combinatorial characterization of (proper and/or improper) learnability of binary classification by computable learners?

Also, it remains intriguing whether the lack of computability requirements was the crucial component in establishing the impossibility of characterizing general (EMAX) learnability by a combinatorial parameter (Ben-David et al., 2017, 2019). Namely:

Open Question Is there a combinatorial characterization of EMAX learnability by computable learners?

In Section 2 below we review the CPAC setup and in Section 3 we provide further background to the problem posed here.

2. Setup

2.1. General background

Computability Let $\Sigma = \{0, 1\}$ be a binary alphabet and let Σ^* be the set of all finite words over Σ. Note that we can naturally identify Σ^* with the natural numbers $\mathbb{N}$ or with the set of all finite subsets of natural numbers. We will often implicitly assume that we fixed one such encoding.

We further assume that we fix some programming language and thus use the existence of Turing machines synonymously with the existence of some *program* or *algorithm* (in our fixed language). A function $f : \Sigma^* \to \Sigma^*$ is said to be *computable* if there exists a program P that halts on every input $\sigma \in \Sigma^*$ and we have $P(\sigma) = f(\sigma)$ for every $\sigma \in \Sigma^*$. A subset S of Σ^* is called *recursively enumerable (RE)* if there exists a program P that takes natural numbers as input, halts on every input and whose range is S. We call a set $S \subseteq \Sigma^*$ *decidable* if there exists a program P that halts on every input $\sigma \in \Sigma^*$ and outputs 1 if $\sigma \in S$ and outputs 0 otherwise.

Learning We let $X = \mathbb{N}$ denote the domain and $Y = \{0, 1\}$ denote the label space. A *hypothesis* is a function $h : X \to Y$. We will often identify such binary functions h with the subset of the domain that h maps to 1 and denote this as $X_h = h^{-1}(1)$. A *hypothesis class* $\mathcal{H} \subseteq Y^X$ is a set of hypotheses. As is common in learning theory, we assume that data is generated i.i.d. by some distribution D over $X \times \{0, 1\}$. We denote the *error* of a hypothesis h with respect to the distribution D by $L_D(h) = \text{Prob}_{(x,y)\sim D}[h(x) \neq y]$. A *learner* is a function that takes in a finite sequence of labeled domain points $S = ((x_1, y_1), \ldots, (x_n, y_n))$ and outputs a hypothesis h.

The standard notion of PAC learnability reflects a guarantee for success from finite sample sizes that holds uniformly for all functions in the hypothesis class $\mathcal{H}$ (and uniformly overall data-generating distributions) (Valiant, 1984; Haussler, 1992). We next review the extension of the framework to computable learners with computable outputs, namely the CPAC framework.

2.2. CPAC learnability

While the CPAC framework does not impose any (polynomial) efficiency requirements on the the runtime of our learners, it requires the learners as well as the output hypotheses to be computable. It is noteworthy that defining a notion of "computable class of functions " is not straightforward (see Remark 4 in Agarwal et al. (2020)), as computability is defined as a property of sets of finite words, while binary classifiers are infinite objects – functions from $\mathbb{N}$ to $\{0, 1\}$. More rigorously, the classic reductions from the Halting problem show that simplicity of the functions in a set of functions does not imply that the *set of programs that encode these simple functions* is decidable: Even for a set containing only one simplest function, say the constant-zero function, the set of all programs that encode it (in any fixed given programming language) is not decidable.

Thus, the CPAC framework considers the following two computability requirements for representations of classes of computable functions (Definitions 5 and 6 in Agarwal et al. (2020)).

Definition 1 (Decidable Representation (DR) of a Hypothesis class) *We say that a class of functions, $\mathcal{H}$, is* Decidably Representable (DR) *if there exists a decidable set of programs $\mathcal{P}$ such that the set of all functions computed by a program in $\mathcal{P}$ is equal to $\mathcal{H}$.*

Definition 2 (Recursively Enumerable Representation (RER) of a Hypothesis class) *We say that a class of functions $\mathcal{H}$ is Recursively Enumerably Representable (RER) if there exists a recursively enumerable set of programs $\mathcal{P}$ such that the set of all functions computed by a program in $\mathcal{P}$ is equal to $\mathcal{H}$.*

A minimal reasonable requirement on the output of a CPAC learner is that it uses a representation that allows for evaluating the output hypothesis on every input of the domain. For example, for a class of functions that have constant output except on a finite subset of their domain, one may also represent the functions by a list of input/output pairs on their effective domain (and one entry

of the output on the rest of the domain), and a learner may output that list. Note that classes that are encodable in this way are not necessarily DR or RER however.

Computable PAC learnability, or *CPAC learnability* for short, is defined as follow (Definition 7 in Agarwal et al. (2020)).

Definition 3 (CPAC learnability) *We say that a class $\mathcal{H}$ is* (agnostic) CPAC *learnable, if there is a computable (agnostic) PAC learner for $\mathcal{H}$ that uses a representation for the predictors it outputs, which allows for evaluating the outputted function on each domain point. If the learner always outputs a (representation of) a hypothesis in class $\mathcal{H}$, we call it a* proper CPAC learner *and the class* proper CPAC learnable.

In addition, one may require that the CPAC learner uses representations according to the definitions of DR and RER classes.

3. Open Questions

We will now give a brief summary of the results in Agarwal et al. (2020) and detail the questions that still remain as open questions from this work. In Agarwal et al. (2020) proper CPAC learning was analysed. It was shown there are classes with finite VC dimension that cannot be properly CPAC learned in the realizable case. When requiring the hypothesis class to be recursively enumerable, however, we get a distinction between the agnostic and the realizable case: Any recursively enumerable class of finite VC dimension is proper CPAC learnable in the realizable case, but there exist decidable hypothesis classes of finite VC dimension that are not proper agnostically CPAC learnable. Whether hypothesis classes of finite VC dimension exist which are not improperly CPAC learnable in the agnostic case remains an open question. The following tables summarize the results from Agarwal et al. (2020) for proper learning and how they transfer to the improper case, serving as an illustration of the remaining open cases. All Theorem numbers refer to Theorems therein.

Proper Learning

	Any class	RE class	DR class
Realizable	PAC $\not\Rightarrow$ CPAC Theorem 9	PAC $\Rightarrow$ CPAC Theorem 10	PAC $\Rightarrow$ CPAC implied by Theorem 10
Agnostic	PAC $\not\Rightarrow$ CPAC implied Theorem 11	PAC $\not\Rightarrow$ CPAC implied by Theorem 11	PAC $\not\Rightarrow$ CPAC Theorem 11

Improper Learning

	Any class	RE class	DR class
Realizable	open	PAC $\Rightarrow$ CPAC implied by Theorem 10	PAC $\Rightarrow$ CPAC implied by Theorem 10
Agnostic	open	open	open

3.1. Some initial steps and conjectures from Agarwal et al. (2020)

We here review some insights and conjectures about improper learning from Agarwal et al. (2020). In the next subsection we then provide a description of a candidate hypothesis which is decidable and has finite VC dimension and which we conjecture to not be improperly CPAC learnable in the agnostic case.

To understand the implications of CPAC-learnability in the more general setup of non-proper learners, it may be useful to consider the following reduction inspired by a reduction in Daniely et al. (2014). For a hypothesis class $\mathcal{H}$ and a sequence $S = ((x_1, y_1), (x_2, y_2), \ldots (x_m, y_m))$ of labeled domain points, we say that S has a *realizable labeling* if there is an $h \in \mathcal{H}$ such that $y_i = h(x_i)$ of all pairs $(x_i, y_i) \in S$. We say that S has a *random labeling* if each y_i was the result of a random coin flip where $\text{Prob}(y_i = 1) = \text{Prob}(y_i = 0) = \frac{1}{2}$. The following task description uses this terminology:

Definition 4 (The Distinguishing Problem) *Given a hypothesis class $\mathcal{H}$, we say that a (potentially randomized) function $A : \bigcup_{n\in\mathbb{N}}(X \times \{0,1\})^n \to \{$"realizable", "unrealizable"$\}$ solves* the distinguishing problem *if for any $\delta > 0$, there is an $M \in \mathbb{N}$ such that for any sequence $S = (x_1, \ldots, x_M) \in X^M$, we have*

- *for T - a realizable labeling of S, $Pr(A(T) =$ "realizable"$) \geq 1 - \delta$,*
- *for T - a random labeling of S, $Pr(A(T) =$ "unrealizable"$) \geq 1 - \delta$,*

where the probability is taken over the randomization of A and the random coin flips that generated the labels in the latter case.

Theorem 5 (Theorem 22 from Agarwal et al. (2020)) *If there is an (agnostic) CPAC learner of a class $\mathcal{H}$ whose range is a class of finite VC-dimension, we can solve the distinguishing problem for $\mathcal{H}$ with a computable distinguisher.*

It is noteworthy that the classes $\mathcal{H}_{halting}$ and $\mathcal{H}_{LT}$ that established the impossibility results for proper CPAC learning in Agarwal et al. (2020), are actually improperly CPAC learnable. All functions in these classes map at most two domain-points to 1 and the rest of the domain to 0. They are thus both subclasses of $\mathcal{H}_2$, the class of all hypotheses h that map at most 2 domain points to label 1. This larger class is actually properly CPAC learnable, which implies that both subclasses $\mathcal{H}_{halting}$ and $\mathcal{H}_{LT}$ are improperly CPAC learnable. This motivates the first conjecture:

Conjecture 6 *If a class $\mathcal{H}$ is (improperly) CPAC learnable, then there is a superclass $\mathcal{H}' \supseteq \mathcal{H}$ such that $\mathcal{H}'$ is proper CPAC learnable.*

Theorem 5, along with Conjecture 6 leads to the following corollary.

Corollary 7 *If $\mathcal{H}$ is (improper) CPAC learnable, then $\mathcal{H}$ is computably distinguishable.*

We also conjecture the following:

Conjecture 8 *There is a class $\mathcal{H}$ consisting of computable hypotheses, and with finite VC-dimension, such that $\mathcal{H}$ is not computably distinguishable.*

Corollary 7, along with Conjecture 8 would imply the existence of a class $\mathcal{H}$ of finite VC-Dimension that is not (improperly) CPAC learnable.

3.2. Candidate hypothesis class

We will now present a candidate hypothesis, which we conjecture not to be CPAC learnable. Let the domain set X be the set of natural numbers, $\mathbb{N}$. Fix a proof system for first order logic over a rich enough vocabulary that is sound and complete (i.e., every first order formula of that language has a proof if and only if it is a logical truth). By "rich enough vocabulary" we mean a finite set of functions symbols and relation symbols so that the set of all its logical truths is undecidable, for example a language for the natural numbers with the ordering, addition, and multiplication. Enumerate all proofs and logical statements of the proof system. We can now define an order $\prec$ over theorems (i.e., statements with a proof), by $i \prec j$ if and only if the proof of i has lower number in the enumeration of proofs than the proof of j. Let

$$h_i(x) = \begin{cases} 1 & \text{if the logical statement with number } x \text{ is a theorem and } x \prec i \\ 0 & \text{otherwise} \end{cases}.$$

We can now define the class $\mathcal{H}_{init} = \{h_i : i \in \mathbb{N}\}$. It is easy to see that the class $\mathcal{H}_{init}$ is decidable and has VC-dimension 1.

Conjecture 9 *The class $\mathcal{H}_{init}$ is not improperly CPAC learnable.*

References

Sushant Agarwal, Nivasini Ananthakrishnan, Shai Ben-David, Tosca Lechner, and Ruth Urner. On learnability wih computable learners. In *Algorithmic Learning Theory, ALT*, pages 48–60, 2020.

Shai Ben-David, Pavel Hrubes, Shay Moran, Amir Shpilka, and Amir Yehudayoff. A learning problem that is independent of the set theory ZFC axioms. *CoRR*, abs/1711.05195, 2017. URL http://arxiv.org/abs/1711.05195.

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

Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Symposium on Theory of Computing, STOC*, pages 441–448, 2014.

David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.

Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.