The Computability of PAC Learning

N. Ackerman[1] **J. Asilis**[1, 2] J. Di[2] C. Freer[3] J. Tristan[2]

[1]Department of Mathematics
Harvard University

[2]Computer Science Department
Boston College

[3]Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology

BC Math & Machine Learning Seminar

Motivation

- Recall the fundamental theorem of machine learning.
  - Perfectly bridges VC theory and PAC learning!

Fundamental Theorem (see, e.g., [SB14, Theorem 6.7])

Let $\mathcal{H}$ be a countable hypothesis class of functions from a domain $\mathcal{X}$ to $\{0, 1\}$. Then the following are equivalent:

1. $\mathcal{H}$ has finite VC dimension.
2. $\mathcal{H}$ is PAC learnable in the realizable case.
3. $\mathcal{H}$ is agnostically PAC learnable.
4. Any ERM learner is an agnostic PAC learner for $\mathcal{H}$.

- To know whether a learner exists, just check the VC dimension.
  - What *exactly* is a learner?

Motivation

- In the fundamental theorem, a learner is a **measurable function** mapping samples to hypotheses.
- Our intention is for computers to do the heavy lifting; learners should (furthermore) be computable!
- What happens if we impose this restriction?
  - How sensitive is the fundamental theorem to computability requirements?
  - How should computable learners even be defined, exactly?

Motivation

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist?
  - Computable proper learner? Computable improper learner?
- If not, any sufficient conditions for computable learners to exist?

Motivation

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist?
  - Computable proper learner? Computable improper learner?
- If not, any sufficient conditions for computable learners to exist?

Not-so-obvious questions:

- What about sample functions?
  - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners?
- Formulating computability in the agnostic vs realizable case? Proper vs improper learning?

Table of Contents

Table of Contents

Computability on $\mathbb{N}$

- Our model of computation is the **Turing machine**.
  - Determined by its program (i.e., transition function, states).
  - Index by natural numbers: $T_1, T_2, \ldots$
  - Can use binary alphabet and implicitly encode input/ouput in binary.

Computability on $\mathbb{N}$

- Our model of computation is the **Turing machine**.
  - Determined by its program (i.e., transition function, states).
  - Index by natural numbers: $T_1, T_2, \ldots$
  - Can use binary alphabet and implicitly encode input/ouput in binary.

Definition

A function $f : \mathbb{N} \to \mathbb{N}$ is **computable** if there exists a Turing machine that halts on each $n \in \mathbb{N}$ and produces $f(n)$ as output. Otherwise, f is **noncomputable**.

- Computable: polynomials, integer division, exponentiation, etc.
  - Proof: We know the algorithms!
- Noncomputable: halting function.
  - $n \mapsto \begin{cases} 1 & T_n \text{ halts on empty input;} \\ 0 & \text{else.} \end{cases}$
  - Proof: Turing's argument . . .

Computability on $\mathbb{N}$

Definition

A set $S \subseteq \mathbb{N}$ is **computable** if its characteristic function is computable. S is **computably enumerable** (c.e.) if it is the range of a computable function (or empty).

- S is computable: one can determine membership in S.
- S is c.e.: one can confirm that $s \in S$ (but not that $s \notin S$).
  - $S = \mathrm{range}(f)$, then compute $f(1), f(2), \ldots$ Any $s \in S$ eventually appears, but when can you conclude $s \notin S$?

Computability on $\mathbb{N}$

Definition

A set $S \subseteq \mathbb{N}$ is **computable** if its characteristic function is computable. S is **computably enumerable** (c.e.) if it is the range of a computable function (or empty).

- S is computable: one can determine membership in S.
- S is c.e.: one can confirm that $s \in S$ (but not that $s \notin S$).
  - $S = \text{range}(f)$, then compute $f(1), f(2), \ldots$ Any $s \in S$ eventually appears, but when can you conclude $s \notin S$?

Lemma

If S is computable, then S is c.e.

- Proof: fix some $s_0 \in S$.

```
def f(n):
    return n if chi_S(n) else s0
```

Example

Let S be finite. Then it is computable (and thus c.e.).

- Proof: hard code the elements of S into your program!

```
def chi_S(x):
    return x in [4, 11, 27, ..., 1034]
```

Computability on $\mathbb{N}$

Example

Let S be finite. Then it is computable (and thus c.e.).

- Proof: hard code the elements of S into your program!

```
def chi_S(x):
    return x in [4, 11, 27, ..., 1034]
```

Example

Let $\mathbf{0}' \subseteq \mathbb{N}$ be the collection of naturals n such that T_n halts on the empty input. Then $\mathbf{0}'$ is not computable.

- Proof idea:

```
def f():
    if halts(f):
        stall()
```

Computability on continuous space

- Want to formalize computation over continuous space, e.g., $\mathbb{R}$.
  - Fundamental obstruction: computers are discrete, $\mathbb{R}$ is uncountable
- Appropriate notion is of an approximation interface: computer requests approximation of input to produce approximation of output.
- Example of such an interface: computable reals

Computability on continuous space

- Want to formalize computation over continuous space, e.g., $\mathbb{R}$.
  - Fundamental obstruction: computers are discrete, $\mathbb{R}$ is uncountable
- Appropriate notion is of an approximation interface: computer requests approximation of input to produce approximation of output.
- Example of such an interface: computable reals

Definition

A **computable real** $x \in \mathbb{R}$ is a real number such that there exists a computable function $f : \mathbb{N} \to \mathbb{Q}$ with $|f(i) - x| < 2^{-i}$.

- Intuition: open balls shrinking around x (and with vibrating center).
- For computable x, y, you can confirm that $x \neq y$ (when true) but never that $x = y$.
  - Fundamental loss in power from discrete case.

Computable metric spaces

- Crucial property of computable reals: *separability* of underlying metric space ($\mathbb{R}$).
    1. Countable: can be input to & output by computers.
    2. Dense: approximate values to arbitrary (but finite) precision.

Computable metric spaces

- Crucial property of computable reals: *separability* of underlying metric space ($\mathbb{R}$).
  1. Countable: can be input to & output by computers.
  2. Dense: approximate values to arbitrary (but finite) precision.

Definition

A **computable metric space** is a triple $\mathbb{X} = (X, d, (s_i)_{i \in \mathbb{N}})$ such that:

1. $(X \cup \{s_i\}_{i \in \mathbb{N}}, d)$ is a separable metric space.
2. $(s_i)_{i \in \mathbb{N}}$, the sequence of **ideal points**, is dense in $(X \cup \{s_i\}_{i \in \mathbb{N}}, d)$.
3. X, the **underlying set** of $\mathbb{X}$, is dense in $(X \cup \{s_i\}_{i \in \mathbb{N}}, d)$.
4. $d(s_i, s_j)$ is a computable real, uniformly in i and j.

- Informally: ideal points play role of $\mathbb{Q}$, computable handle on X.
  - May want underlying set to be $\mathbb{R} \setminus \mathbb{Q}$. Still want $\mathbb{Q}$ as dense subset!

Computable functions on metric spaces

- Things get tedious quickly; don't pay too close attention.

Definition

Let $\mathbb{X}$ and $\mathbb{Y}$ be computable metric spaces with ideal points $(s_i)_{i\in\mathbb{N}}$ and $(t_i)_{i\in\mathbb{N}}$. $f : X \to Y$ is **computable** if for all $(j, q) \in \mathbb{N} \times \mathbb{Q}$ there is a set $\Phi_{j,q} \subseteq \mathbb{N} \times \mathbb{Q}$ such that

- $f^{-1}(B(t_j, q)) = \cup_{(k,p)\in\Phi_{j,q}} B(s_k, p)$, and
- $\{(j, q, k, p) \ : \ (k, p) \in \Phi_{j,q}\}$ is c.e.

- In English: given open ball in codomain, can enumerate the open balls building its pre-image.

Computable functions on metric spaces

- Things get tedious quickly; don't pay too close attention.

Definition

Let $\mathbb{X}$ and $\mathbb{Y}$ be computable metric spaces with ideal points $(s_i)_{i\in\mathbb{N}}$ and $(t_i)_{i\in\mathbb{N}}$. $f : X \to Y$ is **computable** if for all $(j, q) \in \mathbb{N} \times \mathbb{Q}$ there is a set $\Phi_{j,q} \subseteq \mathbb{N} \times \mathbb{Q}$ such that

- $f^{-1}(B(t_j, q)) = \cup_{(k,p)\in\Phi_{j,q}} B(s_k, p)$, and
- $\{(j, q, k, p) \ : \ (k, p) \in \Phi_{j,q}\}$ is c.e.

- In English: given open ball in codomain, can enumerate the open balls building its pre-image.
- Immediate consequence: computable maps are continuous!
  - For maps $\mathbb{N} \to \mathbb{N}$, totally meaningless.
  - For more general spaces, can be quite important . . .

Table of Contents

Computable learning over $\mathbb{N}$

- What should it mean for A to be a 'computable PAC learner' for $\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$?
  - A emits computable functions $\mathbb{N} \to \{0,1\}$.
  - A itself is a computable map $\mathbb{N} \to \mathbb{N}$.
    - Encodings of samples $\mapsto$ encodings of (programs for) functions.
- Simply put: A can be computed and its output can be used to compute predictions.

Computable learning over $\mathbb{N}$

- What should it mean for A to be a 'computable PAC learner' for $\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$?
    - A emits computable functions $\mathbb{N} \to \{0,1\}$.
    - A itself is a computable map $\mathbb{N} \to \mathbb{N}$.
        - Encodings of samples $\mapsto$ encodings of (programs for) functions.
- Simply put: A can be computed and its output can be used to compute predictions.

Definition ([Aga+20, Definition 8])

$\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$ is **computably PAC learnable** if there is a computable PAC learner for $\mathcal{H}$ that outputs code for computable functions.

Conditions on $\mathcal{H}$ and $\mathbb{D}$

- In classical PAC learning, learnability is formulated with respect to a class of possible distributions $\mathbb{D}$ over $\mathcal{X} \times \mathcal{Y}$.
  - Informally, learner must succeed on any distribution $\mathcal{D} \in \mathbb{D}$.
- Two important choices of $\mathbb{D}$:

Conditions on $\mathcal{H}$ and $\mathbb{D}$

- In classical PAC learning, learnability is formulated with respect to a class of possible distributions $\mathbb{D}$ over $\mathcal{X} \times \mathcal{Y}$.
  - Informally, learner must succeed on any distribution $\mathcal{D} \in \mathbb{D}$.
- Two important choices of $\mathbb{D}$:
  1. *Agnostic PAC learning*: $\mathbb{D}$ consists of all Borel distributions on $\mathcal{X} \times \mathcal{Y}$.
  2. *Realizable PAC learning*: $\mathbb{D}$ consists of distributions for which some $h \in \mathcal{H}$ attains true error of 0.
     - Some $h \in \mathcal{H}$ is the true labeling function!

Conditions on $\mathcal{H}$ and $\mathbb{D}$

- In classical PAC learning, learnability is formulated with respect to a class of possible distributions $\mathbb{D}$ over $\mathcal{X} \times \mathcal{Y}$.
  - Informally, learner must succeed on any distribution $\mathcal{D} \in \mathbb{D}$.
- Two important choices of $\mathbb{D}$:
  1. *Agnostic PAC learning*: $\mathbb{D}$ consists of all Borel distributions on $\mathcal{X} \times \mathcal{Y}$.
  2. *Realizable PAC learning*: $\mathbb{D}$ consists of distributions for which some $h \in \mathcal{H}$ attains true error of 0.
     - Some $h \in \mathcal{H}$ is the true labeling function!
- For computable learning, also valuable to consider computability conditions on $\mathcal{H}$.

Definition ([Aga+20, Definition 6])

$\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$ is **computably enumerably representable** (CER) if there exists a c.e. set of programs P such that the set of functions computed by a program in P equals $\mathcal{H}$.

Computable enumerability and ERM

Theorem ([Aga+20, Theorem 10])

Let $\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$ be a CER class. Then an empirical risk minimization (ERM) learner for $\mathcal{H}$ is computable in the realizable case.

Proof.

Let $(h_i)_{i\in\mathbb{N}}$ be a computable enumeration of $\mathcal{H}$, and fix a sample S in the graph of some h_j. In particular, $L_S(h_j) = 0$. Then an $h_k \in L_S^{-1}(0)$ can be found by iterating through $\mathcal{H}$ and calculating empirical error. □

Corollary

Let $\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$ be a CER class of finite VC dimension. Then $\mathcal{H}$ is computably PAC learnable in the realizable case.

Computability makes proper learning strictly harder

Theorem ([Aga+20, Theorem 9])

There is a hypothesis class of VC dimension 1 that does not have any proper computable PAC learners (even in the realizable case).

Proof.

Let $h_i(x) = \begin{cases} 1 & x = 2i; \\ 1 & x = 2i + 1 \text{ and } i \in \mathbf{0}'; \\ 0 & \text{else.} \end{cases}$

Set $\mathcal{H}_{\mathrm{halt}} = \{h_i\}_{i \in \mathbb{N}}$, and suppose A is a proper PAC learner in the realizable case. Then you can compute $\mathbf{0}'$ from A as follows. Fix $n \in \mathbb{N}$, and train A on samples of the form $S = \big((2n, 1), \ldots, (2n, 1)\big)$. Eventually $A(S) = h_n$, exactly when $A(S)(2n) = 1$. Then compute $h_n(2n + 1) = \chi_{\mathbf{0}'}(n)$, as desired. □

Computability makes proper learning strictly harder

Corollary

There is a hypothesis class of finite VC dimension whose proper PAC learners are all noncomputable.

- The fundamental theorem fails under computability constraints!
  - Class of finite VC dimension (1!) without computable proper learners.
- Lesson from $\mathcal{H}_{\mathrm{halt}}$: fork in the road.
  1. Consider improper learners.
     - Does fundamental theorem hold for improper learning?
     - $\mathcal{H}_{\mathrm{halt}}$ has a computable improper PAC learner!
  2. Demand mild conditions (e.g., CER) to prevent classes from memorizing noncomputable sets.
     - Hopefully classes encountered ‘in nature’ do not have this problem ...

Table of Contents

Learners on metric spaces

- Binary classification over computable metric space $\mathcal{X}$.
- First attempt: computable learner should be a computable map $(\mathcal{X} \times \mathcal{Y})^{<\omega} \to \mathcal{Y}^{\mathcal{X}}$.
  - LHS and RHS thought of as computable metric spaces.
  - Obstruction: $\mathcal{Y}^{\mathcal{X}}$ is not in general a computable metric space!

Learners on metric spaces

- Binary classification over computable metric space $\mathcal{X}$.
- First attempt: computable learner should be a computable map $(\mathcal{X} \times \mathcal{Y})^{<\omega} \to \mathcal{Y}^{\mathcal{X}}$.
  - LHS and RHS thought of as computable metric spaces.
  - Obstruction: $\mathcal{Y}^{\mathcal{X}}$ is not in general a computable metric space!
- Instead, *curry* the definition of a learner.

Definition ([Ack+21, Definition 2.18])

A **learner** is a Borel measurable function $A\colon (\mathcal{X} \times \mathcal{Y})^{<\omega} \times \mathcal{X} \to \mathcal{Y}$. A **computable learner** is a learner that is computable as a map of computable metric spaces.

- Extend PAC learning criterion to such learners by simply uncurrying.
  - I.e., consider the map $\tilde{A}(S)$ with $\tilde{A}(S)(x) = A(S, x)$.

Computable presentations

- Once again, want to (sometimes) impose basic computability constraints on $\mathcal{H}$.
  - Intuitively, an analogue of CER for the continuous case.
- Identify elements of $\mathcal{H}$ using an index space.

Computable presentations

- Once again, want to (sometimes) impose basic computability constraints on $\mathcal{H}$.
  - Intuitively, an analogue of CER for the continuous case.
- Identify elements of $\mathcal{H}$ using an index space.

Definition ([Ack+21, Definition 3.2])

A **presentation of a hypothesis class** is a Borel measurable function $\mathfrak{H} \colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$. We call $\mathcal{I}$ the **index space**, and write $\mathfrak{H}^\dagger$ for the underlying hypothesis class, i.e., $\mathrm{range}(i \in \mathcal{I} \mapsto \mathfrak{H}(i, \cdot))$

Computable presentations

- Once again, want to (sometimes) impose basic computability constraints on $\mathcal{H}$.
  - Intuitively, an analogue of CER for the continuous case.
- Identify elements of $\mathcal{H}$ using an index space.

Definition ([Ack+21, Definition 3.2])

A **presentation of a hypothesis class** is a Borel measurable function $\mathfrak{H}\colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$. We call $\mathcal{I}$ the **index space**, and write $\mathfrak{H}^\dagger$ for the underlying hypothesis class, i.e., $\mathrm{range}(i \in \mathcal{I} \mapsto \mathfrak{H}(i, \cdot))$

Definition ([Ack+21, Definition 3.3])

A presentation $\mathfrak{H}\colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$ of a hypothesis class is **computable** if $\mathcal{I}$ is a computable metric space and $\mathfrak{H}$ is computable as a map of computable metric spaces.

- Not so different from CER: 'walk through' $\mathfrak{H}^\dagger$ using ideal points of $\mathcal{I}$.

Proper learning

- Computable presentations set the stage for proper learning.
  - Fix a presentation $\mathfrak{H} \colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$.
  - Learner can output hypotheses in $\mathcal{H}$ (i.e., $\mathfrak{H}^\dagger$) via their indices.
- Proper learners should take advantage of the structure in $\mathcal{I}$!

Proper learning

- Computable presentations set the stage for proper learning.
  - Fix a presentation $\mathfrak{H}\colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$.
  - Learner can output hypotheses in $\mathcal{H}$ (i.e., $\mathfrak{H}^\dagger$) via their indices.
- Proper learners should take advantage of the structure in $\mathcal{I}$!

Definition ([Ack+21, Definition 3.4])

Let $\mathfrak{H}\colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$ be a presentation of a hypothesis class. A **proper learner** for $\mathfrak{H}$ is a map $\mathfrak{A}\colon (\mathcal{X} \times \mathcal{Y})^{<\omega} \to \mathcal{I}$. If the map A defined by

$$A(S, x) = \mathfrak{H}(\mathfrak{A}(S), x)$$

is a PAC learner for $\mathfrak{H}^\dagger$, then $\mathfrak{A}$ is a **proper PAC learner** for $\mathfrak{H}$.

We call A the **learner induced** by $\mathfrak{A}$. If $\mathfrak{H}$ is a computable presentation, then $\mathfrak{A}$ is **computable** when it is computable as a map of computable metric spaces.

Learning in the realizable case

- Consider computable learning in the realizable case for fixed $\mathcal{H}$.
    - Guaranteed that an $h \in \mathcal{H}$ is the true labeling function!
- Computable learner is nevertheless required to be computable on all of $(\mathcal{X} \times \mathcal{Y})^{<\omega} \times \mathcal{X}$.
    - $(\mathcal{X} \times \mathcal{Y})^{<\omega}$ includes samples wildly inconsistent with every $h \in \mathcal{H}$.
    - In realizable case, can ignore such perverse samples.

Learning in the realizable case

- Consider computable learning in the realizable case for fixed $\mathcal{H}$.
  - Guaranteed that an $h \in \mathcal{H}$ is the true labeling function!
- Computable learner is nevertheless required to be computable on all of $(\mathcal{X} \times \mathcal{Y})^{<\omega} \times \mathcal{X}$.
  - $(\mathcal{X} \times \mathcal{Y})^{<\omega}$ includes samples wildly inconsistent with every $h \in \mathcal{H}$.
  - In realizable case, can ignore such perverse samples.

Definition ([Ack+21, Definition 3.5])

For a hypothesis class $\mathcal{H}$, define $\Phi_{\mathcal{H}}$ to be the set of those finite sequences $(x_i, y_i)_{i \in [n]}$ for which $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ lies in the graph of an $h \in \mathcal{H}$.

- Learners in the realizable case need only compute on $\Phi_{\mathcal{H}}$!

Computability in the realizable case

Definition ([Ack+21, Definition 3.6])

A learner A for $\mathcal{H}$ is **computable in the realizable case** if it is computable on $\Phi_{\mathcal{H}} \times \mathcal{X}$. A proper learner for a computable presentation $\mathfrak{H}$ of $\mathcal{H}$ is **computable in the realizable case** if it is computable on $\Phi_{\mathcal{H}}$.

- Weaker notion of computability for learning in the realizable case.
  - Informally, computer is allowed to stall/fail on samples that aren't labeled by an $h \in \mathcal{H}$.
- Learners in the realizable case only need to succeed on $\Phi_{\mathcal{H}}$; they also only need to be computable on $\Phi_{\mathcal{H}}$.

Example: decision stump

- Classical learning problem: decision stump over $\mathbb{R}$.
  - $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$, $\mathcal{H}_{\mathrm{halt}} = \{\mathbf{1}_{>c} \ : \ c \in \mathbb{R}\}$.
- PAC learnable in realizable case with following algorithm:
  1. Set m to be the maximal negatively labeled example (label 0) or minimal positively labeled example (label 1).
  2. Return $\mathbf{1}_{>m}$.

Example: decision stump

- Classical learning problem: decision stump over $\mathbb{R}$.
  - $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$, $\mathcal{H}_{\mathrm{halt}} = \{\mathbf{1}_{>c} \; : \; c \in \mathbb{R}\}$.
- PAC learnable in realizable case with following algorithm:
  1. Set m to be the maximal negatively labeled example (label 0) or minimal positively labeled example (label 1).
  2. Return $\mathbf{1}_{>m}$.
- Problem: classical algorithm is *not* a computable learner
  - $\mathbf{1}_{>m}$ is not computable from S (not even continuous!).
- Topological issue: $\mathbb{R}$ is connected.
  - Only computable maps $\mathbb{R} \to \{0, 1\}$ are constant!
  - Problem must be reformulated ...

Example: decision stump

- New decision stump: $\mathcal{H} = \{\mathbf{1}_{>c} \; : \; c \in \mathbb{R}_c\}$ and $\mathcal{X} = \mathbb{R} \setminus \mathbb{R}_c$ ($\mathbb{R}_c$ = computable reals).
  - $\mathcal{X}$ is totally disconnected.
  - Hypotheses in $\mathcal{H}$ are continuous (and computable!) on $\mathcal{X}$.
    - Cutoff points are not in $\mathcal{X}$.

- New setup even has a computable presentation!

$$\begin{aligned} \mathfrak{H}_{\text{step}} : \mathbb{R}_c \times (\mathbb{R} \setminus \mathbb{R}_c) &\longrightarrow \{0, 1\} \\ c, x &\longmapsto \mathbf{1}_{>c}(x) \end{aligned}$$

Example: decision stump

- New decision stump: $\mathcal{H} = \{\mathbf{1}_{>c} \ : \ c \in \mathbb{R}_c\}$ and $\mathcal{X} = \mathbb{R} \setminus \mathbb{R}_c$ ($\mathbb{R}_c$ = computable reals).
  - $\mathcal{X}$ is totally disconnected.
  - Hypotheses in $\mathcal{H}$ are continuous (and computable!) on $\mathcal{X}$.
    - Cutoff points are not in $\mathcal{X}$.
- New setup even has a computable presentation!

$$\begin{aligned}\mathfrak{H}_{\text{step}} : \mathbb{R}_c \times (\mathbb{R} \setminus \mathbb{R}_c) &\longrightarrow \{0, 1\} \\ c, x &\longmapsto \mathbf{1}_{>c}(x)\end{aligned}$$

- But classical algorithm still fails ...
  - Largest negatively labeled feature m lies in $\mathcal{X}$.
  - So $\mathbf{1}_{>m}$ is discontinuous on $\mathcal{X}$ and noncomputable.

Example: decision stump

- Despite failure of classical algorithm, computable learners exist!

Algorithm $\mathfrak{A}_{\mathrm{step}}$

Fix a computable enumeration $(q_i)_{i\in\mathbb{N}}$ of $\mathbb{Q}$. We define a proper learner $\mathfrak{A}_{\mathrm{step}}$ for $\mathfrak{H}_{\mathrm{step}}$ in the realizable case as follows: given a sample S, output first $q_i \in \mathbb{Q}$ for which the empirical error of $\mathbf{1}_{>q_i}$ is 0.

Example: decision stump

- Despite failure of classical algorithm, computable learners exist!

Algorithm $\mathfrak{A}_{\mathrm{step}}$

Fix a computable enumeration $(q_i)_{i\in\mathbb{N}}$ of $\mathbb{Q}$. We define a proper learner $\mathfrak{A}_{\mathrm{step}}$ for $\mathfrak{H}_{\mathrm{step}}$ in the realizable case as follows: given a sample S, output first $q_i \in \mathbb{Q}$ for which the empirical error of $\mathbf{1}_{>q_i}$ is 0.

- $\mathfrak{A}_{\mathrm{step}}$ is computable in the realizable case.
  - The functions $(\mathbf{1}_{>q_i})_{i\in\mathbb{N}}$ are uniformly computable on $\mathcal{X}$ (as $\mathbb{Q} \subseteq \mathbb{R}_c$).
  - Such a $q_i \in \mathbb{Q}$ is guaranteed to exist as we are in the realizable case.
- $\mathfrak{A}_{\mathrm{step}}$ is a computable proper PAC learner in the realizable case!
  - $\mathfrak{A}_{\mathrm{step}}$ induces an ERM learner on the underlying hypothesis class.
  - Underlying class has VC dimension 1; invoke fundamental theorem.
- In fact, learnability via $\mathfrak{A}_{\mathrm{step}}$ is an instance of a general result ...

Learning computable presentations

Theorem ([Ack+21, Theorem 4.2])

Suppose $\mathfrak{H} \colon \mathcal{I} \times \mathcal{X} \to \mathcal{Y}$ is a computable presentation. Then there is an ERM for $\mathfrak{H}^{\dagger}$ that is computable in the realizable case.

Proof sketch.

We provide a proper learner: search through the ideal points of $\mathcal{I}$, calculating empirical errors, and return the first to attain an error of 0. By continuity of $\mathfrak{H}$ and of empirical error, the collection of $i \in \mathcal{I}$ attaining an error of 0 is an open set. Because we are in the realizable case, the set is furthermore non-empty, thus it contains an ideal point. □

- Generalization of $\mathfrak{A}_{\mathrm{step}}$ and of CER result from the discrete case!

Sample functions

- Suppose we want a procedure for mapping ϵ, δ to a hypothesis with desired error rate and failure probability.
  - Require computable learner *and* computable sample function.
- Do all learners have some computable sample functions? What about computable learners?

Sample functions

- Suppose we want a procedure for mapping ϵ, δ to a hypothesis with desired error rate and failure probability.
  - Require computable learner *and* computable sample function.
- Do all learners have some computable sample functions? What about computable learners?

Theorem ([Ack+21, Theorem 3.12])

There exists a computable PAC learner A for a hypothesis class $\mathcal{H}$ and collection of measures $\mathbb{D}$ such that any sample function for A is noncomputable.

Noncomputable sample functions

Theorem ([Ack+21, Theorem 3.12])

There exists a computable PAC learner whose sample functions are all noncomputable.

Proof sketch.

Let $(e_k)_{k\in\mathbb{N}}$ be a computable enumeration without repetition of $\mathbf{0}'$. Construct a learner A such that $A(S)$ incurs a true error of $1/e_{|S|}$.

Observe that $e_{|S|} > n$ for $|S| > m(1/n, \cdot)$, due to the PAC criterion on m. So $n \in \mathbf{0}'$ if and only if $n \in (e_k)_{k\leq m(1/n,\cdot)}$. Then $\mathbf{0}'$ is computable from a sample function m and the computable enumeration $(e_k)_{k\in\mathbb{N}}$. □

- E.g., to know whether $10 \in \mathbf{0}'$, computing $m(1/10, \cdot) = 300$. Then $10 \notin \mathbf{0}'$ if it does not appear in $(e_k)_{k\leq 300}$.

Table of Contents

Research questions

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist?
  - Computable proper learner? Computable improper learner?
- If not, any sufficient conditions for computable learners to exist?

Not-so-obvious questions:

- What about sample functions?
  - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners?
- Formulating computability in the agnostic vs realizable case?

Research questions

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist?
    - Computable proper learner? No. Computable improper learner?
- If not, any sufficient conditions for computable learners to exist?

Not-so-obvious questions:

- What about sample functions?
    - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners?
- Formulating computability in the agnostic vs realizable case?

Research questions

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist? No.
  - Computable proper learner? No. Computable improper learner?
- If not, any sufficient conditions for computable learners to exist?

Not-so-obvious questions:

- What about sample functions?
  - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners?
- Formulating computability in the agnostic vs realizable case?

Research questions

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist? No.
  - Computable proper learner? No. Computable improper learner?
- If not, any sufficient conditions for computable learners to exist? Computable presentation + realizability.

Not-so-obvious questions:

- What about sample functions?
  - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners?
- Formulating computability in the agnostic vs realizable case?

Research questions

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist? No.
  - Computable proper learner? No. Computable improper learner?
- If not, any sufficient conditions for computable learners to exist? Computable presentation + realizability.

Not-so-obvious questions:

- What about sample functions?
  - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners? No and No.
- Formulating computability in the agnostic vs realizable case?

Research questions

Natural questions:

- Does finite VC dimension suffice for some *computable* ERM learner to exist? No.
  - Computable proper learner? No. Computable improper learner?
- If not, any sufficient conditions for computable learners to exist? Computable presentation + realizability.

Not-so-obvious questions:

- What about sample functions?
  - Do arbitrary PAC learners have computable sample functions? Do computable PAC learners? No and No.
- Formulating computability in the agnostic vs realizable case? Weaken computability restriction using $\Phi_{\mathcal{H}}$.

Fundamental theorem, revisited

Fundamental Theorem (see, e.g., [SB14, Theorem 6.7])

Let $\mathcal{H}$ be a countable hypothesis class of functions from a domain $\mathcal{X}$ to $\{0, 1\}$. Then the following are equivalent:

1. $\mathcal{H}$ has finite VC dimension.
2. $\mathcal{H}$ is PAC learnable in the realizable case.
3. $\mathcal{H}$ is agnostically PAC learnable.
4. Any ERM learner is an agnostic PAC learner for $\mathcal{H}$.

- 1. $\Rightarrow$ 4. fails in the computable setting.
  - Even if you weaken from ERM learners to proper learners!
- Does finite VC dimension guarantee computable *improper* learnability?
  - Open question!
- Non-uniform learning? Computability of multi-label classification? Regression?

References

[Ack+21] Nathanael Ackerman, Julian Asilis, Jieqi Di, Cameron Freer, and Jean-Baptiste Tristan. *On computable learning of continuous features*. 2021. arXiv: 2111.14630 [cs.LG].

[Aga+20] Sushant Agarwal, Nivasini Ananthakrishnan, Shai Ben-David, Tosca Lechner, and Ruth Urner. "On Learnability with Computable Learners". In: *Proceedings of the 31st International Conference on Algorithmic Learning Theory (ALT)*. Vol. 117. PMLR. 2020, pp. 48–60. URL: http://proceedings.mlr.press/v117/agarwal20b.html.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019.