On the Computable Learning of Continuous Features

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

CCA, July 2021

Motivation

Probably Approximately Correct (PAC) learning theory has roughly two 'camps':

1. Efficient PAC learning: learners are computable with polynomial running time.
2. VC theory: learners are arbitrary measurable functions.

Motivation

Probably Approximately Correct (PAC) learning theory has roughly two 'camps':

1. Efficient PAC learning: learners are computable with polynomial running time.
2. VC theory: learners are arbitrary measurable functions.

We look in-between: learners are computable but not resource-bounded. Work so far has focused on the discrete case [Aga+20]; we work over continuous domains as well.

- Are there problems with only noncomputable learners?
- Under what conditions are we guaranteed a computable learner (without complexity constraints)?

Machine learning

- Informal idea: use a set of labeled objects to predict labels on future objects.
- Formally, (feature, label) pairs are drawn from a universe $\mathcal{X} \times \mathcal{Y}$ via an unknown probability distribution $\mathcal{D}$.

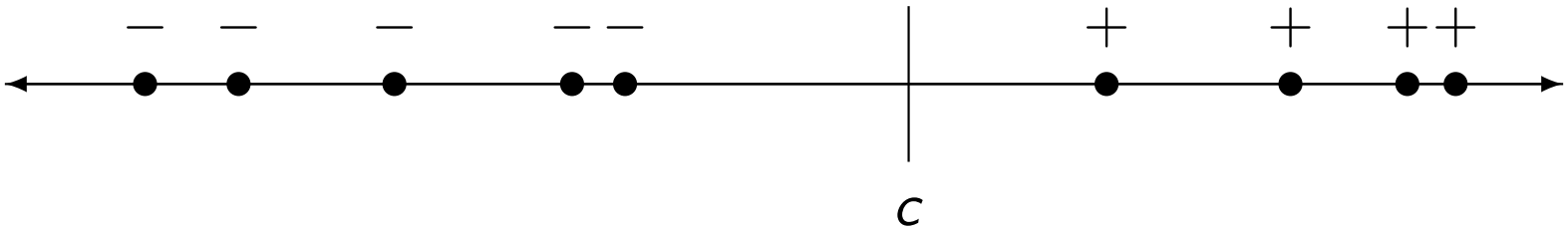
Machine learning

- Informal idea: use a set of labeled objects to predict labels on future objects.
- Formally, (feature, label) pairs are drawn from a universe $\mathcal{X} \times \mathcal{Y}$ via an unknown probability distribution $\mathcal{D}$.
- A *learner* is a map $A\colon (\mathcal{X} \times \mathcal{Y})^{<\omega} \to \mathcal{Y}^{\mathcal{X}}$ that uses a $\mathcal{D}$-i.i.d. sequence $S = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$ to output a *hypothesis* $h\colon \mathcal{X} \to \mathcal{Y}$.
- The goal is for h to have low *true error* $L_{\mathcal{D}}$, i.e.

$$L_{\mathcal{D}}(h) = \mathcal{D}\big(\{(x, y) \mid y \neq h(x)\}\big).$$

Example: Decision Stump

- Binary classification ($\mathcal{Y} = \{0, 1\}$) over real-valued features ($\mathcal{X} = \mathbb{R}$).
- Promised that $\mathcal{D}$ has full measure on $(-\infty, c] \times \{0\} \bigcup (c, \infty) \times \{1\}$ for some c.



- Learning amounts to estimating c.

Learner: Empirical Risk Minimization (ERM)

- Goal of learner is to produce $h\colon \mathcal{X} \to \mathcal{Y}$ with low true error $L_{\mathcal{D}}(h) = \mathcal{D}(\{(x, y) \mid y \neq h(x)\})$.
- Problem: $\mathcal{D}$ is unknown to the learner (that's the point!)
- Proxy for true error is *empirical error* on sample $S = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$, i.e.

$$L_S(h) = \frac{\sum_{i=1}^{n} |h(x_i) - y_i|}{n}.$$

Learner: Empirical Risk Minimization (ERM)

- Goal of learner is to produce $h\colon \mathcal{X} \to \mathcal{Y}$ with low true error $L_{\mathcal{D}}(h) = \mathcal{D}(\{(x, y) \mid y \neq h(x)\})$.
- Problem: $\mathcal{D}$ is unknown to the learner (that's the point!)
- Proxy for true error is *empirical error* on sample $S = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$, i.e.

$$L_S(h) = \frac{\sum_{i=1}^{n} |h(x_i) - y_i|}{n}.$$

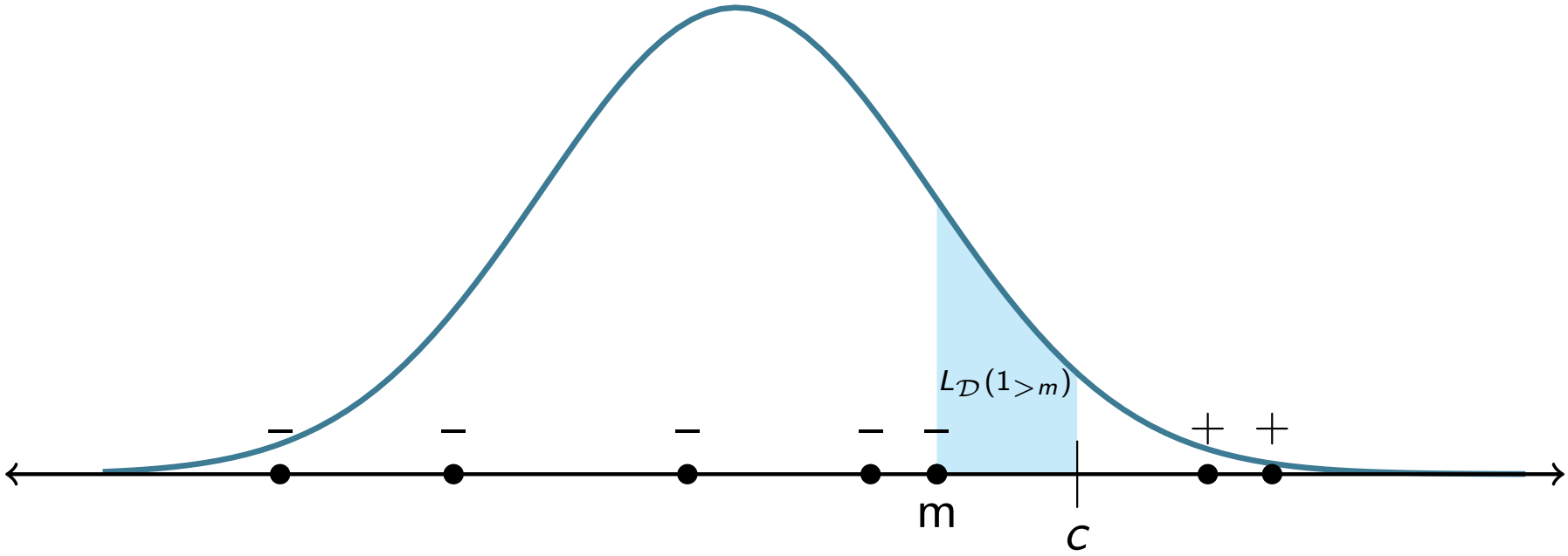
Definition

Fix a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. A learner A is an **empirical risk minimizer** (ERM) for $\mathcal{H}$ if $A(S) \in \arg\min_{h \in \mathcal{H}} L_S(h)$ for all $S \in (\mathcal{X} \times \mathcal{Y})^{<\omega}$.

- We will later provide sufficient conditions for an ERM learner to be computable.

Example: Decision Stump

- Data generating process: feature x carries negative label iff $x < c$ for some fixed c (up to set of measure zero).
- There is an ERM learner A defined by $S \mapsto 1_{>m}$, for m the maximal negatively labeled feature in S.



PAC Learning

- Need a notion of 'success' for learners. The most appropriate is with respect to a hypothesis class $\mathcal{H}$.
  - Informally, if A learns $\mathcal{H}$, then the output of A should be nearly as good as the 'best' hypothesis in $\mathcal{H}$ as $|S| \to \infty$.
- Success will also depend on the class of allowable distributions.
  - Often one assumes the data generating process to take a certain form (e.g., as in the decision stump).
- Primary framework is that of *Probably Approximately Correct* (PAC) learning.

PAC Learning

Definition

Let $\mathbb{D}$ be a collection of distributions on $\mathcal{X} \times \mathcal{Y}$ and let $\mathcal{H}$ be a class of hypotheses. A learner A is said to **learn $\mathcal{H}$ with respect to $\mathbb{D}$** if there exists a function $m\colon (0,1)^2 \to \mathbb{N}$ such that for all:

- $\epsilon, \delta \in (0,1)$
- $\mathcal{D} \in \mathbb{D}$
- $n \geq m(\epsilon, \delta)$

If A is trained on a $\mathcal{D}$-i.i.d. sample S of size n, then

$$L_{\mathcal{D}}(A(S)) \leq \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

with probability at least $(1 - \delta)$ over the choice of S. When such an A exists, we say that $\mathcal{H}$ is **PAC learnable with respect to $\mathbb{D}$**.

- We will later define a notion of computable PAC learnability.

PAC Learning: Agnostic and Realizable

- Informally: A learns $\mathcal{H}$ with respect to $\mathbb{D}$ if A picks $h \in \mathcal{H}$ whose error decreases with increasing probability as $|S| \to \infty$ (for any $\mathcal{D} \in \mathbb{D}$).
- Two important choices of $\mathbb{D}$:
  1. *Agnostic PAC learning*: $\mathbb{D}$ consists of all distributions on $\mathcal{X} \times \mathcal{Y}$.
  2. *Realizable PAC learning*: $\mathbb{D}$ consists of those distributions $\mathcal{D}$ for which $L_{\mathcal{D}}(h) = 0$ for some $h \in \mathcal{H}$.
- In the diagram, we considered learning $\mathcal{H}_{\text{step}} = \{1_{>r} \mid r \in \mathbb{R}\}$ in the realizable case.

Fundamental Theorem of Statistical Learning

- Agnostic PAC learning is the most stringent goal.
  - An agnostic PAC learner for $\mathcal{H}$ is a PAC learner for $\mathcal{H}$ over any other collection of distributions.
- Fundamental Theorem establishes surprising equivalence between the agnostic and realizable cases, when learners need not be computable.
  - Also establishes a connection with finiteness of the VC-dimension (which we will not need).

Fundamental Theorem of Statistical Learning

- Agnostic PAC learning is the most stringent goal.
  - An agnostic PAC learner for $\mathcal{H}$ is a PAC learner for $\mathcal{H}$ over any other collection of distributions.
- Fundamental Theorem establishes surprising equivalence between the agnostic and realizable cases, when learners need not be computable.
  - Also establishes a connection with finiteness of the VC-dimension (which we will not need).

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. $\mathcal{H}$ is PAC learnable in the realizable case.
2. $\mathcal{H}$ is agnostically PAC learnable.
3. Any ERM learner is an agnostic PAC learner for $\mathcal{H}$.

Computable Learning

- Fundamental theorem allows learners to be arbitrary measurable functions, possibly noncomputable.
- We demand that learners be computable.
- Similar framework has been studied by [Aga+20] with $\mathcal{X} = \mathbb{N}$, $\mathcal{Y} = \{0, 1\}$. We permit $\mathcal{X}$ to be an arbitrary *computable Polish space*.
  - Also in [Cro+21] with slightly different spaces and maps.
- Key notion, defined in [Aga+20], is that of a *computably enumerable representable* (CER) hypothesis class, i.e. a class that admits a computable enumeration of codes for its elements.

Computable PAC Learning

Definition

A CER hypothesis class $\mathcal{H}$ is **computably agnostically PAC learnable** if there is an agnostic PAC learner A for $\mathcal{H}$ which is computable as a function from $(\mathcal{X} \times \mathcal{Y})^{<\omega}$ to $\mathcal{H}$, considered as metric spaces.

The CER class $\mathcal{H}$ is **computably PAC learnable in the realizable case** if there is a PAC learner A for $\mathcal{H}$ in the realizable case which is computable on the set of finite sequences $\big((x_1, y_1), \ldots, (x_n, y_n)\big)$ for which $\big\{(x_1, y_1), \ldots, (x_n, y_n)\big\}$ is a subset of the graph of some $h \in \mathcal{H}$.

Results

Write $\mathbf{0}'$ for the halting set, i.e., the set of $n \in \mathbb{N}$ for which the nth Turing machine halts on empty input. Throughout, $\mathcal{Y} = \{0, 1\}$ and $\mathcal{X}$ is a computable Polish space.

Theorem 1

For every CER class $\mathcal{H}$ that is PAC learnable, some ERM learner of it is $\mathbf{0}'$-computable.

Theorem 2

There is a PAC learnable CER hypothesis class $\mathcal{H}$ such that every ERM learner for it computes $\mathbf{0}'$.

Theorem 3

For every CER class $\mathcal{H}$ that is PAC learnable, some ERM learner of it is computable in the realizable case, and hence $\mathcal{H}$ is computably PAC learnable in the realizable case.

References

[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. Proceedings of Machine Learning Research. San Diego, California, USA, 2020, pp. 48–60.

[Cro+21] Tonicha Crook, Jay Morgan, Arno Pauly, and Markus Roggenbach. "A Computability Perspective on (Verified) Machine Learning". In: *arXiv e-print 2102.06585* (2021). eprint: 2102.06585. URL: https://arxiv.org/abs/2102.06585.

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

Theorem 1

Theorem

For every CER class $\mathcal{H}$ that is PAC learnable, some ERM learner of it is $\mathbf{0}'$-computable.

Proof sketch

Let $(h_i)_{i\in\mathbb{N}}$ be a computable enumeration of $\mathcal{H}$, and fix a sample S. For each $i \in \mathbb{N}$, there is a program that loops over $j \in \mathbb{N}$ searching for one with $L_S(h_j) < L_S(h_i)$.

Using a halting oracle, we can determine whether such a j exists. If it exists, we can again find $k \in \mathbb{N}$ so that $L_S(h_k) < L_S(h_j)$, and so on. This terminates because L_S takes values in $\{0, \frac{1}{|S|}, \frac{2}{|S|}, \ldots, 1\}$.

Theorem 2

Theorem

There is a PAC learnable CER hypothesis class $\mathcal{H}$ such that every ERM learner for it computes $\mathbf{0}'$.

Proof sketch

Set $\mathcal{X} = \mathbb{N}$ and $\mathcal{H} = \{\chi_n \mid n \in \mathbf{0}'\}$, i.e. $\chi_n(x) = \begin{cases} 1 & x = n; \\ 0 & \text{else.} \end{cases}$

Say A is an ERM learner for $\mathcal{H}$. Then $A\big(((n,1))\big) = \chi_n$ iff $n \in \mathbf{0}'$. So $\chi_{\mathbf{0}'}$ can be computed via

$$n \mapsto A\big(((n,1))\big)(n).$$

Theorem 3

Theorem

For every CER class $\mathcal{H}$ that is PAC learnable, some ERM learner of it is computable in the realizable case. In particular, $\mathcal{H}$ is computably PAC learnable in the realizable case.

Proof sketch

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 (computably) found by iterating through $\mathcal{H}$ and calculating empirical risk.

This describes an ERM algorithm, which is a PAC learner by the fundamental theorem.