

Lecture 2: Probably Approximately Correct Learning

Tianpei Xie

Jul. 30th., 2015

Contents

1	PAC Learning for Realizable Case	2
1.1	PAC Learning Definitions	2
1.2	PAC-Learnable Guarantees for Finite Hypothesis Sets	3
1.3	PAC-Learnable Examples for Infinite Hypothesis Sets	5
2	Agnostic PAC-Learning	5
2.1	Non-Realizable Case	6
3	PAC Learning vs. Universal Consistency	7

1 PAC Learning for Realizable Case

1.1 PAC Learning Definitions

- **Remark** (*Settings*)

In *deterministic scenario*, denote a collection of n *independent identically distributed (i.i.d.) random samples* generated by P_X as $X = (X_1, \dots, X_n)$, and *label* is generated from a concept $c \in \mathcal{C}$ as $Y_i = c(X_i)$. Denote the sample as

$$\mathcal{D} \equiv \mathcal{D}_n = ((X_1, Y_1), \dots, (X_n, Y_n)) \equiv ((X_1, c(X_1)), \dots, (X_n, c(X_n))).$$

Thus, $\mathcal{D}_n|_X \sim \mathcal{P}_X^n$ is a random sequence of n i.i.d. samples.

A **learner** \mathcal{A} considers a **fixed subset of concepts** $\mathcal{H} \subset \mathcal{C}$, which is referred as a **hypothesis class**, and provides a **hypothesis** or a **classifier** or a **decision function** $h \in \mathcal{H}$. The task of **supervised learning** is to minimize the *generalization error*:

$$L(h) \equiv L_{\mathcal{P},c}(h) = \mathcal{P}_X \{h(X) \neq c(X)\} \equiv \mathbb{E}_{\mathcal{P}} [\mathbb{1}_{h(X) \neq c(X)}]$$

where $\mathbb{1}_{\omega}$ is the indicator function of the event ω .

In particular, a *learning algorithm (such as ERM)* \mathcal{A} takes sample \mathcal{D}_n as input and output

$$\mathcal{A}(\mathcal{D}_n) = h_n = h_{\mathcal{D}_n}^*(x) \equiv h(x|((X_1, Y_1), \dots, (X_n, Y_n))).$$

Given that \mathcal{D}_n are random samples, $h_n \equiv h_{\mathcal{D}_n}$ **are random** as well. Thus **the generalization error** of h_n is a **random variable**:

$$L_n \equiv L_{\mathcal{P},c}(h_n) := \mathbb{E} [\mathbb{1} \{h_{\mathcal{D}_n}(X) \neq c(X)\} | \mathcal{D}_n] = \mathcal{P} \{h_{\mathcal{D}_n}(X) \neq c(X) | \mathcal{D}_n\}.$$

- **Remark** We denote by $\mathcal{O}(d)$ an *upper bound on the cost of the computational representation* of any element $x \in \mathcal{X} \subset \mathbb{R}^d$ and by $\text{size}(c)$ **the maximal cost of the computational representation of** $c \in \mathcal{C}$. For example, x may be a vector in \mathbb{R}^d , for which the cost of an array-based representation would be in $\mathcal{O}(n)$.

- **Remark** (***Realizability Assumption***)

There exists a hypothesis $h \in \mathcal{H}$ such that **the generalization error is zero**.

$$\exists h \in \mathcal{H}, \quad L_{\mathcal{P},c}(h) = 0$$

Every ERM hypothesis has zero training error. In the definition of ***Probably Approximately Correct (PAC) Learning***, we assume the realizability assumption holds.

- **Definition** (***Probably Approximately Correct (PAC) Learning, Realizable Case***)

Assume that *the realizability assumption holds*. A hypothesis class \mathcal{H} is said to be **Probably Approximately Correct-learnable** i.e. **PAC-learnable**, if there exists an *algorithm* \mathcal{A} and a **polynomial function** $\text{poly}(\cdot, \cdot, \cdot, \cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for **all distributions** \mathcal{P} on \mathcal{X} and for any target concept $c \in \mathcal{C}$, the following holds when the **sample size** $n \geq \text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$:

$$\mathcal{P} \{L_{\mathcal{P},c}(\mathcal{A}(\mathcal{D}_n)) \leq \epsilon\} \geq 1 - \delta. \quad (1)$$

If \mathcal{A} further runs in $\text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$, then \mathcal{H} is said to be **efficiently PAC-learnable**. When such an algorithm \mathcal{A} exists, it is called a **PAC-learning algorithm** for \mathcal{H} .

- **Remark** A hypothesis class \mathcal{H} is thus **PAC-learnable** if the hypothesis returned by the algorithm after observing a number of points **polynomial** in $1/\epsilon$ and $1/\delta$ is

1. **approximately correct** (error at most ϵ)
2. **with high probability** (at least $1 - \delta$),

which justifies the PAC terminology.

1. $\delta > 0$ is used to define **the confidence** $1 - \delta$
2. and $\epsilon > 0$ **the accuracy** $1 - \epsilon$.

Note that if the running time of the algorithm is **polynomial** in $1/\epsilon$ and $1/\delta$, then **the sample size** n must also be polynomial if the full sample is received by the algorithm.

- **Remark (Confidence and Accuracy)**

1. For a fixed classifier h , the **accuracy** of that classifier is measured by $1 - L_{\mathcal{P},c}(h)$.
2. The **confidence** of learning algorithm is measured by **the probability of failure** for hypothesis returned by the learning algorithm. A failure happens when the learned classifier h has **low accuracy**, i.e. $L_{\mathcal{P},c}(h) > \epsilon$. Since classifier $h_{\mathcal{D}}$ is function of data \mathcal{D}_n , the **confidence** is measured by **the probability of encountering bad samples** \mathcal{D}_n such that $\hat{L}_{\mathcal{D}}(h_{\mathcal{D}}) = 0$ but $L_{\mathcal{P},c}(h_{\mathcal{D}}) > \epsilon$.

\mathcal{H} is **PAC-learnable** when a learning algorithm has **high confidence** to obtain a classifier $h \in \mathcal{H}$ with **high accuracy regardless** of the underlying **data distribution** and **labeling logic**.

- **Remark** Several key points of the PAC definition are worth emphasizing.

1. First, the PAC framework is a **distribution-free**: no particular assumption is made about the distribution \mathcal{P} from which examples are drawn.
2. Second, the training sample and the test examples used to define the error are drawn according to **the same distribution** \mathcal{P} . This is a necessary assumption for generalization to be possible in most cases.
3. Third, the PAC framework is **non-asymptotic** in which it provides a **concentration inequality** that holds for all n .
4. Finally, the PAC framework deals with the question of **learnability for a hypothesis class** \mathcal{H} and not a particular hypothesis h . Note that **the concept class** \mathcal{C} is known to the algorithm, but of course target concept $c \in \mathcal{C}$ is unknown.

We may omit the polynomial dependency on n and $\text{size}(c)$ in the PAC definition and focus only on the **sample complexity**.

1.2 PAC-Learnable Guarantees for Finite Hypothesis Sets

- **Remark (Finite Hypothesis Sets)**

We will see if the size of hypothesis class \mathcal{H} is finite, i.e. $|\mathcal{H}| < \infty$, we can achieve PAC learnability under the realizability assumption:

- **Proposition 1.1** (*Learning bounds, Finite \mathcal{H} , Realizable Case*) [Mohri et al., 2018]
Let \mathcal{H} be a **finite** set of functions mapping from \mathcal{X} to \mathcal{Y} . Let \mathcal{A} be an algorithm that for **any target concept** $c \in \mathcal{H}$ and i.i.d. sample \mathcal{D}_n returns a consistent hypothesis h_n , i.e. **the training error of h_n is zero**: $\widehat{L}(h_n) = 0$. Then, for any $\epsilon, \delta > 0$, the inequality

$$\mathcal{P}^n \{L_{\mathcal{P},c}(h_n) \leq \epsilon\} \geq 1 - \delta$$

holds if

$$n \geq \frac{1}{\epsilon} \log \left(\frac{|\mathcal{H}|}{\delta} \right) \quad (2)$$

This sample complexity result admits the following **equivalent statement** as a generalization bound: for any $\epsilon, \delta > 0$, with **probability** at least $1 - \delta$,

$$L_n \equiv L_{\mathcal{P},c}(h_n) \leq \frac{1}{n} \log \left(\frac{|\mathcal{H}|}{\delta} \right). \quad (3)$$

Proof: We want to upper bound the probability of event

$$M := \left\{ \mathcal{D}_n \sim \mathcal{P}^n : \exists h \in \mathcal{H}, L_{\mathcal{P},c}(h) > \epsilon \wedge \widehat{L}(h) = 0 \right\}.$$

In other word, for each sample $\mathcal{D}_n \in M$, there are bad hypothesis that seems good in the training data \mathcal{D}_n . Under the realizability assumption, all samples that makes $L_{\mathcal{P},c}(h_n) > \epsilon$ lies in M . Moreover, M can be rewritten as

$$M = \bigcup_{h \in \mathcal{H}_B} \left\{ \mathcal{D}_n \sim \mathcal{P}^n : \widehat{L}(h) = 0 \right\}$$

where $\mathcal{H}_B := \{h \in \mathcal{H} : L_{\mathcal{P},c}(h) > \epsilon\}$. Then by union bound,

$$\begin{aligned} \mathcal{P}^n \{L_{\mathcal{P},c}(h_{\mathcal{D}_n}) > \epsilon\} &\leq \mathcal{P}^n \left\{ \bigcup_{h \in \mathcal{H}_B} \left\{ \mathcal{D}_n : \widehat{L}(h) = 0 \right\} \right\} \\ &\leq \sum_{h \in \mathcal{H}_B} \mathcal{P}^n \left\{ \mathcal{D}_n : \widehat{L}(h) = 0 \right\} \end{aligned} \quad (4)$$

Note that

$$\begin{aligned} \mathcal{P}^n \left\{ \mathcal{D}_n : \widehat{L}(h) = 0 \right\} &= \mathcal{P}^n \{h(X_i) = c(X_i), i = 1, \dots, n\} \\ &= \prod_{i=1}^n \mathcal{P} \{h(X_i) = c(X_i)\} \end{aligned}$$

For each individual sampling of an element of the training set we have

$$\mathcal{P} \{h(X_i) = c(X_i)\} = 1 - L_{\mathcal{P},c}(h) \leq 1 - \epsilon$$

where the last inequality follows from the fact that $h \in \mathcal{H}_B$. Thus, for all $h \in \mathcal{H}_B$,

$$\mathcal{P}^n \left\{ \mathcal{D}_n : \widehat{L}(h) = 0 \right\} \leq (1 - \epsilon)^n \leq \exp(-n\epsilon)$$

where the last inequality comes from $1 - x \leq e^{-x}$. Combining this equation with (4) we conclude that

$$\mathcal{P}^n \{L_{\mathcal{P},c}(h_{\mathcal{D}_n}) > \epsilon\} \leq |\mathcal{H}_B| \exp(-n\epsilon) \leq |\mathcal{H}| \exp(-n\epsilon).$$

Setting the right-hand side to be equal to δ and solving for ϵ concludes the proof. ■

- **Example** (*Conjunction of Boolean Literals*) [Mohri et al., 2018]

1.3 PAC-Learnable Examples for Infinite Hypothesis Sets

- **Example** (*Learning Axis-Aligned Rectangles*) [Mohri et al., 2018]

Consider the case where the set of instances are *points* in the plane, $\mathcal{X} = \mathbb{R}^2$, and the concept class \mathcal{C} is the set of **all axis-aligned rectangles** lying in \mathbb{R}^2 . Thus, each concept c is *the set of points inside a particular axis-aligned rectangle*. The learning problem consists of determining with small error a target axis-aligned rectangle using the labeled training sample. We will show that *the concept class of axis-aligned rectangles is **PAC-learnable***.

- **Example** (*Threshold Function Class is Learnable*) [Shalev-Shwartz and Ben-David, 2014, Mohri et al., 2018]

Let \mathcal{H} be the set of threshold functions over the *real line*, namely,

$$\mathcal{H} = \{h_\alpha : \alpha \in \mathbb{R}\}, \quad \text{where } h_\alpha(x) = \mathbb{1}_{\{x < \alpha\}}.$$

Clearly, \mathcal{H} is of infinite size.

Nevertheless, the following lemma shows that \mathcal{H} is *learnable* in the PAC model using the ERM algorithm.

Lemma 1.2 [Shalev-Shwartz and Ben-David, 2014]

Let \mathcal{H} be the class of **thresholds** as defined earlier. Then, \mathcal{H} is **PAC learnable**, using the ERM rule, with **sample complexity** of

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \log \left(\frac{2/\delta}{\epsilon} \right) \right\rceil.$$

2 Agnostic PAC-Learning

- **Definition** (*Agnostic PAC-Learning*)

Let \mathcal{H} be a hypothesis set. \mathcal{A} is an **agnostic PAC-learning algorithm** if there exists a **polynomial function** $\text{poly}(\cdot, \cdot, \cdot, \cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions \mathcal{P} over $\mathcal{X} \times \mathcal{Y}$, the following holds for any sample size $n \geq \text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$:

$$\mathcal{P} \left\{ L_{\mathcal{P}}(\mathcal{A}(\mathcal{D}_n)) - \inf_{h \in \mathcal{H}} L(h) \leq \epsilon \right\} \geq 1 - \delta. \quad (5)$$

If \mathcal{A} further runs in $\text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$, then \mathcal{C} is said to be **efficiently agnostic PAC-learnable**.

- **Remark** (*Agnostic PAC-Learning \Leftarrow Uniformly Control of the Estimation Error*)
The difference between *the error of a hypothesis* $h \in \mathcal{H}$ and *the Bayes error* can be decomposed as:

$$L(h) - L^* = \underbrace{\left(L(h) - \inf_{h \in \mathcal{H}} L(h) \right)}_{\text{estimation error}} + \underbrace{\left(\inf_{h \in \mathcal{H}} L(h) - L^* \right)}_{\text{approximation error}}.$$

where the first term is the **estimation error** and the second term is the **approximation error**.

- When \mathcal{H} and \mathcal{P} are fixed, the approximation error is fixed.
- The definition of **agnostic PAC learnability** requires that the estimation error would be bounded above uniformly over all distributions.

$$\begin{aligned} L(h_n) - \inf_{h \in \mathcal{H}} L(h) &= L(h_n) - \widehat{L}(h_n) + \widehat{L}(h_n) - L(h^*) \\ &\leq L(h_n) - \widehat{L}(h_n) + \widehat{L}(h^*) - L(h^*) \\ &\leq 2 \sup_{h \in \mathcal{H}} |L(h) - \widehat{L}(h)| \end{aligned} \quad (6)$$

where $h^* = \operatorname{argmin}_{h \in \mathcal{H}} L(h)$ and $\widehat{L}(h)$ is the training error of g . The second last inequality is due to the fact that h_n minimizes the training error. Thus the estimation error can be bounded uniformly by the generalization error bound $|L(h) - \widehat{L}(h)|$ for any $h \in \mathcal{H}$.

- **Remark** From the inequality (6), we see that the excess risk is **uniformly bounded** by the deviation of generation error from the training error.

$$\sup_{h \in \mathcal{H}} |\widehat{L}(h) - L(h)| := \|\widehat{\mathcal{P}}_n - \mathcal{P}\|_{\mathcal{H}}$$

Definition (Glivenko-Cantelli Class) [Wainwright, 2019, Giné and Nickl, 2021]

We say that \mathcal{H} is a **Glivenko-Cantelli class** for \mathcal{P} if $\|\widehat{\mathcal{P}}_n - \mathcal{P}\|_{\mathcal{H}}$ converges to zero in probability as $n \rightarrow \infty$.

2.1 Non-Realizable Case

- To prove the general case, we use a bound from *Hoeffding's inequality*:

Proposition 2.1 Fix $\epsilon > 0$ and let \mathcal{D}_n denote an i.i.d. sample of size n . Then, for any hypothesis $h : \mathcal{X} \rightarrow \{0, 1\}$, the following inequalities hold:

$$\mathcal{P}^n \left\{ \widehat{L}(h) - L(h) \geq \epsilon \right\} \leq \exp(-2n\epsilon^2) \quad (7)$$

$$\mathcal{P}^n \left\{ \widehat{L}(h) - L(h) \leq -\epsilon \right\} \leq \exp(-2n\epsilon^2) \quad (8)$$

By the union bound, this implies the following two-sided inequality:

$$\mathcal{P}^n \left\{ |\widehat{L}(h) - L(h)| \geq \epsilon \right\} \leq 2 \exp(-2n\epsilon^2). \quad (9)$$

Setting the right-hand side of (9) to be equal to δ and solving for ϵ yields immediately the following bound for a single hypothesis

Corollary 2.2 (Generalization bound for Single Hypothesis) [Mohri et al., 2018]

Fix a hypothesis $h : \mathcal{X} \rightarrow \{0, 1\}$. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:

$$L(h) \leq \widehat{L}(h) + \sqrt{\frac{\log \frac{2}{\delta}}{2n}} \quad (10)$$

This error bound can be seen as coming from *the randomness of coin tossing* when approximate the generalization error $L(h)$ by training error. Thus it will always exist for any generalization error bound.

- **Proposition 2.3** (*Learning bounds, Finite \mathcal{H} , Non-Realizable Case*) [Mohri et al., 2018]

Let \mathcal{H} be a **finite** set of functions mapping from \mathcal{X} to \mathcal{Y} . Then, for any $\delta > 0$ with probability at least $1 - \delta$, the following inequality holds: for all $h \in \mathcal{H}$

$$L(h) \leq \hat{L}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{2}{\delta}}{2m}} \quad (11)$$

Thus for a finite hypothesis set \mathcal{H} ,

$$L(h) \leq \hat{L}(h) + O\left(\sqrt{\frac{\log |\mathcal{H}|}{m}}\right) \quad (12)$$

- **Remark** (*Sample Efficiency \Leftarrow Representation Efficiency of Hypothesis Class*)
As already pointed out, $\log |\mathcal{H}|$ can be interpreted as *the number of bits needed to represent \mathcal{H}* . Several other remarks similar to those made on the generalization bound in the consistent case can be made here:

a larger sample size m guarantees **better generalization**, and the bound **increases** with $|\mathcal{H}|$, but only **logarithmically**.

But, here, the bound is a less favorable function of $\log |\mathcal{H}| / m$; it varies as *the square root of this term*. This is not a minor price to pay: for a *fixed* $|\mathcal{H}|$, to attain the *same guarantee as in the consistent case*, a **quadratically larger labeled sample** is needed.

- **Example** (*k-term DNF Formulae*) [Mohri et al., 2018]
- **Example** (*k-CNF Formulae*) [Mohri et al., 2018]

3 PAC Learning vs. Universal Consistency

- **Remark** (*Universal Consistency is Not Enough*) [Shalev-Shwartz and Ben-David, 2014]
We compare the *universal consistency* and *PAC learnability* as **performance guarantee** for a classification rule:

1. **The universal consistency** of a classification rule provides a *performance guarantee* in terms of **asymptotic analysis**. It concerns that if *the generalization error* of classification rule $\{h_n\}$ will reach to *infimum* within given class \mathcal{H} when the sample size is **infinity** $n \rightarrow \infty$. **The universal consistency** is a *measurable of correctness*, i.e. the classification rule can reach to *correct solution eventually* regardless of the underlying distribution of data.

On the other hand, it does not answer *how many examples are required to be as good as the best hypothesis in \mathcal{H}* . The answer to this question depends on the underlying

distribution $\mathcal{P}_{X,Y}$ in consistency statement. *The consistency statement is more related to **large sample statistical behavior**.*

2. **The PAC learnability** of a classification rule provides a *performance guarantee* in terms of **non-asymptotic analysis**. It provides a *measure of efficiency*. An **algorithm** is *efficient* if it obtains a solution close to *correct solution* (within an error rate ϵ) with high probability $(1 - \delta)$ **given a finite set of $n \geq n(\epsilon, \delta)$ samples**.

The PAC learnability is a measure for the **algorithm** as well as **the statistical nature** of the problem. Its formulation is closer to *computer science* than to *statistics*.

Moreover, the definition of PAC learning yields **the limitation** of learning (via the **No-Free-Lunch theorem**) and **the necessity of prior knowledge**. It gives us a crisp way to encode prior knowledge by choosing a *hypothesis class*, and once this choice is made, we have a generic learning rule - *ERM*. Compared to PAC learning, the definition of **consistency** does **not** yield a natural learning paradigm or a way to **encode prior knowledge**. In fact, in many cases there is *no need for prior knowledge at all* since the consistency only cares about the *asymptotic behavior*.

Consider the classification prediction algorithm **Memorize** defined as follows. The algorithm *memorizes* the training examples, and, given a test point x , it predicts the majority label among all labeled instances of x that exist in the training sample (and some fixed default label if no instance of x appears in the training set). It is possible to show (see Exercise below) that the **Memorize** algorithm is **universally consistent** for **every countable domain \mathcal{X}** and a **finite label set \mathcal{Y}** (w.r.t. the zero-one loss).

Intuitively, it is *not obvious* that the **Memorize** algorithm should be viewed as a *learner*, since it **lacks** the aspect of **generalization**, namely, of using observed data to *predict* the labels of unseen examples. The fact that **Memorize** is a *consistent algorithm* for the class of all functions over any countable domain set therefore raises *doubt* about **the usefulness of consistency guarantees**. Furthermore, the sharp-eyed reader may notice that the “*bad learner*” we introduced in Chapter 2, which led to *overfitting*, is in fact the **Memorize** algorithm.

• **Exercise 3.1 (Memorize Algorithm)**

In this example we wish to show that the algorithm **Memorize** is a **consistent learner** for **every class of (binary-valued) functions over any countable domain**.

Let \mathcal{X} be a **countable domain** and let \mathcal{P} be a probability distribution over \mathcal{X} .

1. Let $\{x_i : i \in \mathbb{N}\}$ be an enumeration of the elements of \mathcal{X} so that for all $i \leq j$, $\mathcal{P}(\{x_i\}) \leq \mathcal{P}(x_j)$. Prove that

$$\lim_{n \rightarrow \infty} \sum_{i \geq n} \mathcal{P}(\{x_i\}) = 0.$$

Note that $\sum_{i=0}^{\infty} \mathcal{P}(\{x_i\}) = 1$ by definition of probability measure.

2. Given any $\epsilon > 0$, prove that there exists $\epsilon_{\mathcal{P}} > 0$ such that

$$\mathcal{P}\{x \in \mathcal{X} : \mathcal{P}(\{x\}) < \epsilon_{\mathcal{P}}\} < \epsilon.$$

3. Prove that for every $\eta > 0$, if n is such that $\mathcal{P}(\{x_i\}) < \eta$ for all $i > n$, then for every

$m \in \mathbb{N}$, let \mathcal{D}_n be the sample of size m generated according to \mathcal{P}

$$\mathcal{P}_{\mathcal{D}_m} \{ \exists x_i : \mathcal{P}(\{x_i\}) > \eta \text{ and } x_i \notin \mathcal{D}_m \} \leq ne^{-\eta m}.$$

4. Conclude that if \mathcal{X} is **countable** then for **every probability distribution** \mathcal{P} over \mathcal{X} there exists a function $m_{\mathcal{P}} : (0, 1) \times (0, 1) \rightarrow \mathbb{N}$ such that for every $\epsilon, \delta > 0$ if $m > m_{\mathcal{P}}(\epsilon, \delta)$ then

$$\mathcal{P}_{\mathcal{D}_m} \{ \mathcal{P}(\{x : x \notin \mathcal{D}_m\}) > \epsilon \} < \delta.$$

5. Prove that **Memorize** is a consistent learner for every class of (binary-valued) functions over any countable domain.

- **Remark (Universal Consistency May Not be Good Preference)** [Shalev-Shwartz and Ben-David, 2014]

One may argue that even though *consistency* is a weak requirement, it is desirable that a learning algorithm will be **consistent** with respect to the set of **all functions** from \mathcal{X} to \mathcal{Y} , which gives us a *guarantee* that for enough training examples, we will always be as good as the Bayes optimal predictor. Therefore, if we have two algorithms, where one is *consistent* and the other one is *not consistent*, we should *prefer* the consistent algorithm. **However**, this argument is **problematic** for two reasons.

1. First, maybe it is the case that for most “*natural*” distributions we will observe in practice that **the sample complexity of the consistent algorithm** will be *so large* so that in every practical situation we will *not obtain enough examples* to enjoy this guarantee.
2. Second, it is **not very hard to make any PAC or nonuniform learner consistent with respect to the class of all functions from \mathcal{X} to \mathcal{Y} .**

Concretely, consider a countable domain, \mathcal{X} , a finite label set \mathcal{Y} , and a hypothesis class, \mathcal{H} , of functions from \mathcal{X} to \mathcal{Y} . We can make any nonuniform learner for \mathcal{H} be consistent with respect to the class of all classifiers from \mathcal{X} to \mathcal{Y} using the following simple trick: Upon receiving a training set, we will first run the nonuniform learner over the training set, and then we will obtain a *bound* on the true risk of the learned predictor. If this bound is *small enough* we are done. Otherwise, we revert to the *Memorize* algorithm. This simple modification makes the algorithm consistent with respect to all functions from \mathcal{X} to \mathcal{Y} .

Since **it is easy to make any algorithm consistent**, it may **not be wise** to **prefer** one algorithm over the other just because of consistency considerations.

References

- Evarist Giné and Richard Nickl. *Mathematical foundations of infinite-dimensional statistical models*. Cambridge university press, 2021.
- Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.
- Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.