UC Berkeley **Jonathan Shafer**
CS 294-220, Spring 2021

Unit 2

Probably Approximately Correct: A Probabilistic Definition of Learning

1 The PAC Definition

In this section we present the Probably Approximately Correct (PAC) definition of learning. We will see a few other definitions throughout the course, but this definition (and a variant of it called *agnostic* PAC) will be our central definition.

PAC is a *probabilistic* definition of learning. Two main motivations for taking a probabilistic approach are that this allows us to:

- Formally make inferences in which "absence of evidence is evidence of absence," in the sense discussed in the previous unit.
- Model learning in the presence of uncertainly and noise.

Additionally, the PAC definition allows us to model learning with finite amounts of data and computational resources (instead of infinite positive presentations, and infinite computations that only converge in the limit).

1.1 Components of a Learning Problem

In a *learning problem*, a learner is attempts predict the specific *labels* $y \in \mathcal{Y}$ that correspond to specific *instances* $x \in \mathcal{X}$. For instance, given measurements of the temperature and humidity today (which together constitute a single instance), predict whether it will rain tomorrow (this the label).

More fully, the learner is attempting to learn an unknown function $f : \mathcal{X} \to \mathcal{Y}$. To do so, it is given example input-output pairs $(x, f(x))$, where x is chosen randomly from $\mathcal{X}$. The setting includes the following components:

- $\mathcal{X}$ – a set called the *domain* or the *instance space*.
- $\mathcal{Y}$ – a set called the *co-domain* or the *label space*.
- The unknown system:
  - $f : \mathcal{X} \to \mathcal{Y}$ – a *target function*.
  - $\mathcal{D}$ – a distribution over $\mathcal{X}$.
- S – a random variable called the *sample* or the *training set*. This is the data presented to the learner. There are a number of reasonable ways in which the data might be generated, but we will mostly consider the following setting: S is a tuple of m ordered pairs $\Big((x_1, y_1), \ldots, (x_m, y_m)\Big)$, such that for every $i \in [m]$, $x_i \in \mathcal{X}$ is sampled independently according to $\mathcal{D}$, and $y_i = f(x_i) \in \mathcal{Y}$. The assumption that the x_i's are sampled independently from $\mathcal{D}$ is called *the i.i.d. assumption*.
- Learner – a (possibly randomized) algorithm that takes S as input and produces a *hypothesis function* $h : \mathcal{X} \to \mathcal{Y}$ as output. Also called a *learning algorithm*.
- $L_{\mathcal{D},f}(h)$ – a *loss function* that measures how well the hypothesis h is able to predict the labels assigned by f with respect to the distribution $\mathcal{D}$. We will generally assume

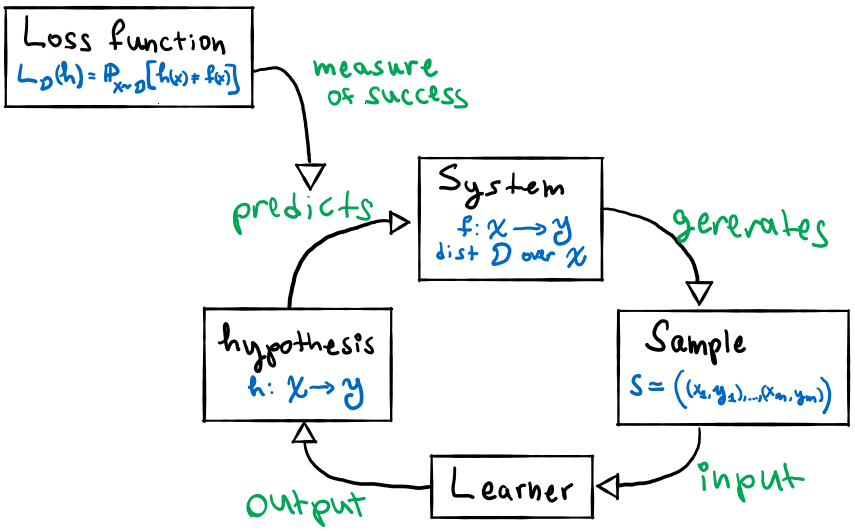

Figure 1 Components of the PAC model.

that $L_{\mathcal{D},f}(h) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\ell(f(x), h(x))]$ for some function ℓ. In particular, we will mostly focus on the '0-1 loss,' where $\ell(y, y') = \mathbb{1}(y \neq y')$, which yields

$$L_{\mathcal{D},f}(h) = \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)].$$

However, there are many other reasonable choices for a loss functions. For example, if $\mathcal{Y} = \mathbb{R}$ we can choose $\ell(f, h, x) = (f(x) \neq h(x))^2$, which yields the expected square error loss, $L_{\mathcal{D},f}(h) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[(h(x) - f(x))^2\right]$.

All details of the setting are known to the learner except for the target function f and the distribution $\mathcal{D}$. The learner's objective is to output a hypothesis h such that $L_{\mathcal{D},f}(h)$ is as small as possible.

▶ Remark 1.

- The unknown distribution $\mathcal{D}$ plays a double role: it determines how the sample S is generated, and it also defines the loss function $L_{\mathcal{D},f}$ used to evaluate the performance of the learner.
- The current setting does not meet our objective of dealing with uncertainty (e.g., noisy labels). We have made a fairly strong assumption about the unknown system – that the label y is a *deterministic function* of x. Next lecture we will see how this assumption can be relaxed.

1.2 PAC Definition – First Attempt

We want the learner to find a hypothesis h that has loss as small as possible. It might seem desirable to require that the learner find h such that $L_{\mathcal{D},f}(h) = 0$, namely $h = f$ on all points in the support of $\mathcal{D}$. However, there are two issues with this:

- The learner only gets a limited amount of information about the target function f, because it gets a sample that only contains a finite number m of input-output pairs. Therefore we generally cannot hope for the learner to reconstruct f precisely. Instead, we can go for the next best thing, which is to require that $L_{\mathcal{D},f}(h) \leq \varepsilon$ for some small positive ε. That is, we require that h be *approximately correct*.

Moreover, we can make $\varepsilon > 0$ arbitrarily small. The more data the learner gets, the better its predictions can be. That is, we require that for every $\varepsilon > 0$ there exists $m \in \mathbb{N}$ such that if the algorithm receives a sample of size m then it will output h such that $L_{\mathcal{D},f}(h) \leq \varepsilon$. (Basically, $L_{\mathcal{D},f}(h) \to 0$ as m goes to infinity, with upper bounds on $L_{\mathcal{D},f}(h)$ also for finite sample sizes m and not only in the limit.)

- Seeing as the sample is generated randomly according to $\mathcal{D}$, there is always a possibility that the learner will be unlucky and get a 'bad' sample. For instance, there is a nonzero probability that the sample $S = \Big((x_1, y_1), \ldots, (x_m, y_m)\Big)$ will have $x_1 = x_2 = \cdots = x_m$, and so the learner only receives information on a single input-output pair, even though the sample is of size $m > 0$. Therefore, we cannot hope for the learner to *always* achieve low loss. However, because the probability of getting a bad sample is low, we can require that it will *probably* succeed, i.e., the learner achieve low loss with high probability.

Combining these two considerations, we get the *probably approximately correct* (PAC) definition of learning. Following is a first (problematic) attempt at formalizing this notion.

▶ **Definition 2** (PAC Learning – ***Naïve Version***)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets. We say that a (possibly randomized) algorithm A is a naïve probably approximately correct learner for functions $\mathcal{X} \to \mathcal{Y}$ if for any precision parameter $\varepsilon \in (0,1)$ and confidence parameter $\delta \in (0,1)$ there exists a sample size $m \in \mathbb{N}$ such that for every target function $f : \mathcal{X} \to \mathcal{Y}$ and every distribution $\mathcal{D}$ over $\mathcal{X}$, if A receives as input the parameters ε and δ and a sample S such that $S = \Big((x_1, f(x_1)), \ldots, (x_m, f(x_m))\Big)$ where $x_1, \ldots, x_m$ are sampled independently from $\mathcal{D}$, then A halts and outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ that with probability at least $1 - \delta$ has loss $L_{\mathcal{D},f}(h) \leq \varepsilon$.*

▶ **Notation 3.** Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, $\mathcal{D}$ be a distribution on $\mathcal{X}$ and $f : \mathcal{X} \to \mathcal{Y}$ be a function. We write $z \sim (\mathcal{D}, f)$ to denote that z is a random variable such that $z = (x, y)$ where $x \in \mathcal{X}$ is sampled according to $\mathcal{D}$ and $y = f(x)$.

The requirement of naïve PAC learning can be summarized concisely as

$$\mathbb{P}_{S \sim (\mathcal{D},f)^m}\left[L_{\mathcal{D},f}(h) \leq \varepsilon\right] \geq 1 - \delta, \tag{1}$$

where $h = A(\varepsilon, \delta, S)$, and the probability is over the randomness of the sample and of the random coins used by the algorithm A (if A is a randomized algorithm).

The number m in the definition is a number of examples that depend on ε and δ and is sufficient to guarantee that the algorithm satisfies (1).

▶ **Definition 4.** *Let A be an algorithm that is a naïve PAC learner for functions $\mathcal{X} \to \mathcal{Y}$, for some sets $\mathcal{X}, \mathcal{Y}$. The sample complexity of A is a function $m : (0,1)^2 \to \mathbb{N}$ such that for every $\varepsilon, \delta \in (0,1)$, the number $m(\varepsilon, \delta)$ is the minimal sample size for which A satisfies (1).*

Let's see what we can say about this naïve definition. As an example, observe that if $\mathcal{X}$ is finite then learning is possible according to Definition 2.

▷ Claim 5. Let $\mathcal{X}$ and $\mathcal{Y}$ be nonempty sets and assume that $\mathcal{X}$ is finite. There exists an algorithm A that naïve PAC learns the functions $\mathcal{X} \to \mathcal{Y}$ and has sample complexity $m(\varepsilon, \delta) \leq \left\lceil \frac{|\mathcal{X}| + \ln(1/\delta)}{\varepsilon} \right\rceil$. ⌟

The learning algorithm simply 'memorizes' the sample. Namely, given a sample $S = \Big((x_1, y_1), \ldots, (x_m, y_m)\Big)$, it outputs the following hypothesis h. For every $x \in \mathcal{X}$, if there

exists $i \in [m]$ such that $x = x_i$, then $h(x) = y_{i^*}$ where $i^* = \min\{i \in [m] : \ x_i = x\}$. Otherwise, $h(x) = y'$ for some arbitrary fixed $y' \in \mathcal{Y}$.

The idea of the proof is simple: If we take a number of samples that is linear in $|\mathcal{X}|$, then we will see the correct labels for nearly all of $\mathcal{X}$, except possibly some small fraction of weight at most ε, and so by memorizing these labels we achieve loss less than ε.

Proof of Claim 5. Fix $\varepsilon > 0$ and $\delta > 0$, a distribution D, and a target function f. We show that if the memorization algorithm described in the previous paragraph receives a sample S of size $m = \left\lceil \frac{|\mathcal{X}|+\ln(1/\delta)}{\varepsilon} \right\rceil$, then it outputs h such that $\mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[L_{\mathcal{D},f}(h) > \varepsilon\right] < \delta$.

We say that a set $X \subseteq \mathcal{X}$ is *bad* if $\mathcal{D}(X) > \varepsilon$ and $\forall x \in X : \ h(x) \neq f(x)$. Note that $L_{\mathcal{D},f}(h) = \mathbb{P}_{x\sim\mathcal{D}}\left[h(x) \neq f(x)\right] = \mathcal{D}(\{x \in \mathcal{X} : h(x) \neq f(x)\})$, so

$$L_{\mathcal{D},f}(h) > \varepsilon \iff \exists X \subseteq \mathcal{X} : \ X \text{ is Bad.}$$

Hence,

$$\begin{aligned}
\mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[L_{\mathcal{D},f}(h) > \varepsilon\right] &= \mathbb{P}_S\left[\exists X \subseteq \mathcal{X} : \ X \text{ is Bad}\right] \\
&= \mathbb{P}_S\left[\bigcup_{X\subseteq\mathcal{X}} \{X \text{ is Bad}\}\right] \\
&\leq \sum_{X\subseteq\mathcal{X}} \mathbb{P}_S\left[X \text{ is Bad}\right]. && \text{(union bound)}
\end{aligned}$$

Let $S_x = \{x_i : \ i \in [m]\} \subseteq \mathcal{X}$ and observe that if $X \cap S_x \neq \varnothing$ then X is not bad, because if $x \in S_x$ then h will memorize the correct label for x and then $h(x) = f(x)$.

Next, we show that for any $X \subseteq \mathcal{X}$, it holds that $\mathbb{P}_S\left[X \text{ is Bad}\right] < \frac{\delta}{2^{|\mathcal{X}|}}$. This suffices to complete the proof, because it implies that $\sum_{X\subseteq\mathcal{X}} \mathbb{P}_S\left[X \text{ is Bad}\right] < \delta$. Notice that for any $X \subseteq \mathcal{X}$, if $\mathcal{D}(X) \leq \varepsilon$ then X is never bad, and if $\mathcal{D}(X) > \varepsilon$ then

$$\begin{aligned}
\mathop{\mathbb{P}}_{S\sim(\mathcal{D},f)^m}\left[X \text{ is Bad}\right] &\leq \mathbb{P}_S\left[X \cap S_x = \varnothing\right] \\
&= \mathbb{P}_S\left[\bigcap_{i\in[m]} \{x_i \notin X\}\right] = \prod_{i\in[m]} \mathbb{P}_S\left[x_i \notin X\right] && (x_i\text{'s are i.i.d.}) \\
&\leq \prod_{i\in[m]} (1-\varepsilon) = (1-\varepsilon)^m \leq e^{-\varepsilon m}.
\end{aligned}$$

But

$$e^{-\varepsilon m} < \frac{\delta}{2^{|\mathcal{X}|}} \iff m > -\frac{\ln\left(\frac{\delta}{2^{|\mathcal{X}|}}\right)}{\varepsilon} = \frac{|\mathcal{X}|\ln(2) + \ln\left(\frac{1}{\delta}\right)}{\varepsilon},$$

and so taking $m = \left\lceil \frac{|\mathcal{X}|+\ln(1/\delta)}{\varepsilon} \right\rceil > \frac{|\mathcal{X}|\ln(2)+\ln\left(\frac{1}{\delta}\right)}{\varepsilon}$ is sufficient to ensure that $e^{-\varepsilon m} < \frac{\delta}{2^{|\mathcal{X}|}}$, as desired. ◀

The result of Claim 5 is not very surprising. Clearly, if we see the labels for nearly all $x \in \mathcal{X}$ then we can perform well on these x's. But this only captures memorization, which is a rudimentary (and quite boring) type of learning. Memorization performs poorly in more realistic situations where the training set contains only a small part of $\mathcal{X}$. Broadly speaking, in learning theory we are more interested in learners that can *generalize* – can use the instances received in the training sample in order to make good predictions about *new*, as yet *unseen* instances not present in the training sample.

▶ **Definition 6.** *Let* $S = \Big((x_1, y_1), \ldots, (x_m, y_m)\Big)$ *be a sample and* $h : \mathcal{X} \to \mathcal{Y}$ *be a hypothesis. The empirical loss of* h *with respect to* S *is* $L_S(h) = \frac{1}{m}\sum_{i\in[m]} \ell(y_i, h(x_i))$*, which in the case of the* 0-1 *loss is*

$$L_S(h) = \frac{1}{m}\sum_{i\in[m]} \mathbb{1}(y_i \neq h(x_i)) = \frac{|\{i \in [m] :\ h(x_i) \neq y_i\}|}{m}.$$

The out-of-sample loss of h *with respect to a distribution* $\mathcal{D}$ *and target function* f *is given by* $\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\ell(f(x), h(x)) \mid x \notin S_x]$*, which for the* 0-1 *loss is*

$$\mathbb{P}_{x\sim\mathcal{D}}[f(x) \neq h(x)) \mid x \notin S_x],$$

where $S_x = \{x_1, \ldots, x_m\}$.

▶ Remark 7. To distinguish $L_{\mathcal{D}}$ from the empirical loss L_S and from the out-of-sample loss, we will sometimes refer to $L_{\mathcal{D}}$ as the *true loss* or the *population loss*.

The memorization algorithm achieves a perfect empirical loss of $L_S(h) = 0$, but its out-of-sample loss is basically the same as what we would expect if we simply assigned labels arbitrarily or by chance – about as poor as possible.

Our goal is to achieve generalization, which can be formalized either as saying that $|L_S(h) - L_{\mathcal{D},f}(h)|$ is small, or as saying that the out-of-sample loss is small. Unfortunately, as we will see in the next section, this is not possible in the general case.

1.3 No Free Lunch

> *"[T]here is nothing in any object, considered in itself, which can afford us a reason for drawing a conclusion beyond it; and [...] even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience;"*
> David Hume, in *A Treatise of Human Nature* (1739), Book I, Part III, Section XII.

In Definition 2 we did not make any assumptions on the target function f – it could be any function whatsoever. Therefore, the values $f(x_i)$ that the learner receives for x_i's in the sample convey no information at all about the value $f(x)$ for any x not in the sample. Therefore, neither memorization or any other algorithm can perform better than chance on out-of-sample instances. Formally:

▶ **Theorem 8** (No Free Lunch, Wolpert [6])**.** *Let* $\mathcal{X}$ *be a nonempty finite set,* $\mathcal{Y} = \{0, 1\}$*, and* $\mathcal{F}$ *be the set of all functions from* $\mathcal{X} \to \mathcal{Y}$*. Let* $\mathcal{U}$ *be the uniform distribution over* $\mathcal{X}$*. Then for any learning algorithm* A *for functions* $\mathcal{X} \to \mathcal{Y}$ *and any* $m \in \mathbb{N}$*,*

$$\frac{1}{2} - \frac{m}{2|\mathcal{X}|} \leq \mathbb{E}_{f\in\mathcal{F}}\ \mathbb{E}_{S\sim(\mathcal{U},f)^m}[L_{\mathcal{U},f}(h)] \leq \frac{1}{2} + \frac{m}{2|\mathcal{X}|},$$

where $h = A(S)$*, and* f *is sampled uniformly from* $\mathcal{F}$*.*

Proof. First, note the following simply case of the law of total expectation. Let A and B be two random variables in a discrete joint probability space, let $g(A, B)$ be a real-valued function, and assume that that $\mathbb{E}[|g(A, B)|] \leq \infty$. Then

$$\begin{aligned}\mathbb{E}[g(A, B)] &= \sum_a \sum_b \mathbb{P}[A = a\ \wedge\ B = b]\, g(a, b) = \sum_a \mathbb{P}[A = a] \sum_b \mathbb{P}[B = b \mid A = a]\, g(a, b)\\ &= \mathbb{E}_A[\mathbb{E}_B[g(A, B) \mid A]].\end{aligned}$$

And so from symmetry, $\mathbb{E}_A\left[\mathbb{E}_B\left[g(A,B) \mid A\right]\right] = \mathbb{E}\left[g(A,B)\right] = \mathbb{E}_B\left[\mathbb{E}_A\left[g(A,B) \mid B\right]\right]$.[1]

Second, define that a sample $S = \Big((x_1,y_1),\ldots,(x_m,y_m)\Big)$ and a function f are *consistent* if $y_i = f(x_i)$ for all $i \in [m]$, and likewise S is *self-consistent* if there exists some function f such that S and f are consistent. Observe that in the theorem we have two random variables f and S in a joint finite probability space, and we are interested in the expectation of the function $g(f,S) = L_{\mathcal{U},f}(A(S))$. The marginal distribution of f is uniform over $\mathcal{F}$, the marginal of S is uniform over all self-consistent samples, and the joint distribution is such that f and S are always consistent.

Third, fix $f \in \mathcal{F}$, and let S_x denote the subset of $\mathcal{X}$ that appears in S. Then

$$\begin{aligned}\mathop{\mathbb{E}}_{S\sim(\mathcal{U},f)^m}\left[L_{\mathcal{U},f}(h)\right] &= \mathop{\mathbb{E}}_{S\sim(\mathcal{U},f)^m}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\mathbb{1}(f(x)\neq h(x))\right] \\ &= \mathop{\mathbb{E}}_{S\sim(\mathcal{U},f)^m}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}\setminus S_x}\mathbb{1}(f(x)\neq h(x))\right] + \mathop{\mathbb{E}}_{S\sim(\mathcal{U},f)^m}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in S_x}\mathbb{1}(f(x)\neq h(x))\right]. \end{aligned} \tag{2}$$

Consider each item in (2) separately when taking an expectation over f. For the first item, invoking the law of total expectation we obtain

$$\mathbb{E}_{f\in\mathcal{F}}\,\mathbb{E}_{S\sim(\mathcal{U},f)^m}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}\setminus S_x}\mathbb{1}(f(x)\neq h(x))\right] = \frac{1}{|\mathcal{X}|}\mathbb{E}_{S\sim(\mathcal{U},f)^m}\left[\sum_{x\in\mathcal{X}\setminus S_x}\mathop{\mathbb{E}}_{f\in\mathcal{F}}\Big[\mathbb{1}(f(x)\neq h(x))\;\Big|\;S\Big]\right]$$

The random variable $f|S$ is uniform over all functions $f \in \mathcal{F}$ that are consistent with S. in particular, for any fixed $x \in \mathcal{X}\setminus S_x$, the value $f(x)$ is uniform over $\mathcal{Y}$, and therefore $\mathbb{E}_{f\in\mathcal{F}}\Big[\mathbb{1}(f(x)\neq h(x))\;\Big|\;S\Big] = \frac{1}{2}$. Hence,

$$\frac{1}{|\mathcal{X}|}\mathbb{E}_{S\sim(\mathcal{U},f)^m}\left[\sum_{x\in\mathcal{X}\setminus S_x}\mathop{\mathbb{E}}_{f\in\mathcal{F}}\Big[\mathbb{1}(f(x)\neq h(x))\;\Big|\;S\Big]\right] = \frac{1}{2}\cdot\frac{|\mathcal{X}|-|S_x|}{|\mathcal{X}|} = \frac{1}{2} - \frac{|S_x|}{2|\mathcal{X}|}.$$

For the second item in (2), note that

$$0 \leq \mathbb{E}_{f\in\mathcal{F}}\,\mathbb{E}_{S\sim(\mathcal{U},f)^m}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in S_x}\mathbb{1}(f(x)\neq h(x))\right] \leq \frac{|S_x|}{|\mathcal{X}|}.$$

Finally, plugging these expressions back into (2) we obtain

$$\frac{1}{2} - \frac{|S_x|}{2|\mathcal{X}|} \leq \mathbb{E}_{S\sim(\mathcal{U},f)^m}\left[L_{\mathcal{U},f}(h)\right] \leq \frac{1}{2} + \frac{|S_x|}{2|\mathcal{X}|}.$$

Noting that $|S_x| \leq m$ completes the proof. ◀

▶ **Exercise 9.** *In the setting of Theorem 8, for any algorithm (like the memorization algorithm) that guarantees that $L_S(h) = 0$, compute an exact expression for $\mathbb{E}_{f\in\mathcal{F}}\,\mathbb{E}_{S\sim(\mathcal{U},f)^m}\left[L_{\mathcal{U},f}(h)\right]$ as a function of m and $|\mathcal{X}|$.*

[1] This is true for general measures by Fubini's Theorem, which states that under mild conditions, $\int_{X\times Y} g(x,y)\mathrm{d}(x,y) = \int_X\left(\int_Y g(x,y)\mathrm{d}y\right)\mathrm{d}x = \int_Y\left(\int_X g(x,y)\mathrm{d}x\right)\mathrm{d}y$.

Hence, without any assumption on the target function f, it doesn't matter which learning algorithm we choose, if the sample size is small relative to $|\mathcal{X}|$ the the expected loss would approximately equal $\frac{1}{2}$, which is the same as it would be if we guessed the labels at random. In particular, as the following corollary shows, naïve PAC learning is not possible unless the sample size is linear in $\mathcal{X}$.

▷ Claim 10 (Markov's Inequality). Let X be an integrable non-negative real-valued random variable. Then for any $t > 0$, $\mathbb{P}[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$.

Proof.

$$\begin{aligned}
\mathbb{P}[X \geq t] &= \mathbb{E}[\mathbb{1}(X \geq t)] && \text{(definition of Lebesgue integral)} \\
&\leq \mathbb{E}\left[\left(\frac{X}{t}\right)\mathbb{1}(X \geq t)\right] && \text{(if } X \geq t > 0 \text{ then } \frac{X}{t} \geq 1\text{)} \\
&= \frac{\mathbb{E}[X \cdot \mathbb{1}(X \geq t)]}{t} \\
&\leq \frac{\mathbb{E}[X]}{t}. && (X \geq 0 \Rightarrow X \geq X\mathbb{1}(X \geq t) \\
& && \Rightarrow \mathbb{E}[X] \geq \mathbb{E}[X\mathbb{1}(X \geq t)])
\end{aligned}$$

◀

▶ **Corollary 11.** *Let $\mathcal{X}$ be a nonempty finite set and $\mathcal{Y} = \{0, 1\}$. The following holds with respect to the uniform distribution $\mathcal{U}$ over $\mathcal{X}$. For any learning algorithm A for functions $\mathcal{X} \to \mathcal{Y}$, there exists a function $f : \mathcal{X} \to \mathcal{Y}$ such that if A receives a sample of size $m \leq \frac{|\mathcal{X}|}{2}$ from $\mathcal{U}$ labeled according to f then A is a poor naïve PAC learner in the sense that*

$$\mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[L_{\mathcal{U},f}(h) \geq \frac{1}{8}\right] \geq \frac{1}{7}.$$

Proof. Fix a learning algorithm A. From Theorem 8 there exists a function f such that

$$\mathbb{E}_{S\sim(\mathcal{U},f)^m}[L_{\mathcal{U},f}(h)] \geq \frac{1}{2} - \frac{m}{2|\mathcal{X}|} \geq \frac{1}{4}$$

For $h = A(S)$. Let q denote the accuracy of h with respect to f, i.e., $q = 1 - L_{\mathcal{U},f}(h)$. From Markov's inequality,

$$\mathbb{P}_{S\sim(\mathcal{U},f)^m}\left[q \geq \frac{7}{8}\right] \leq \frac{8 \cdot \mathbb{E}_S[q]}{7} = \frac{8\left(1 - \mathbb{E}_S[L_{\mathcal{U},f}(h)]\right)}{7} \leq \frac{8\left(1 - \frac{1}{4}\right)}{7} = \frac{6}{7}.$$

In other words,

$$\frac{1}{7} \leq 1 - \mathbb{P}_S\left[q \geq \frac{7}{8}\right] = \mathbb{P}_S\left[q < \frac{7}{8}\right] = \mathbb{P}_{S\sim(\mathcal{U},f)^m}\left[L_{\mathcal{U},f}(h) > \frac{1}{8}\right].$$

◀

Hence, if the domain is infinite, naïve PAC learning is impossible:

▶ **Corollary 12.** *Let $\mathcal{X}$ be an infinite set and $\mathcal{Y} = \{0, 1\}$. Then there does not exist an algorithm that naïve PAC learns the functions $\mathcal{X} \to \mathcal{Y}$.*

Proof. Assume for contradiction that A is a leaning algorithm that naïve PAC learns the functions $\mathcal{X} \to \mathcal{Y}$. Then in particular there exists $m \in \mathbb{N}$ such that for all distributions $\mathcal{D}$ and target functions f, $h = A(S)$ satisfies $\mathbb{P}_{S\sim(\mathcal{D},f)^m}[L_{\mathcal{D},f}(h) \leq 1/10] \geq 9/10$ where S is a sample of size m taken from $\mathcal{D}$ and labeled according to f. Fix $X \subseteq \mathcal{X}$ such that $|X| = 2m$, and let $\mathcal{U}$ be the uniform distribution on X. By Corollary 11, there exists a target function f such that $\mathbb{P}_{S\sim(\mathcal{U},f)^m}[L_{\mathcal{U},f}(h) \geq 1/8] \geq 1/7$, a contradiction to the previous inequality. ◀

The last corollary suggests that the definition of naïve PAC learning might be too strong (be too hard to achieve), because there are many examples in the real world where it appears to be possible to generalize well when learning over an infinite domain. For example, spam filters are fairly good at classifying whether an email is spam or ham, even though the suitable domain $\mathcal{X}$ (the set of all possible emails) is infinite.

1.4 A Better PAC Definition: Incorporating Inductive Bias

In the previous section we presented one possible formalization of Hume's observation, showing that if we do not assume anything about the unknown system then we cannot generalize from instances that appear in the sample to unseen instances. Therefore, it appear that successful generalization requires making more assumptions about the unknown system. These assumptions are called *inductive bias*, because the learner has an a-priori preference for (or bias in favor of) some hypotheses over other hypotheses.

One example is the preference in science for simple hypotheses. All else being equal, scientists will usually prefer a simple hypothesis over a more complex one, even if they both fit the evidence equally well. This bias is known as *Occam's razor*, and we will revisit it later on in the course.

As another example, consider the following experiment conducted by Gracia and Koelling [1] which explored the learning proclivities of rats. They exposed rats to two types of stimuli, an audiovisual stimulus, and a gustatory stimulus (food). For each type of stimulus, they conducted an experiment where they subjected rats to one of three unpleasant conditions: electric shock, a toxin, or x-ray radiation (both the toxin and the radiation cause nausea). They found that the rats can learn to associate:

- The the electric shock with the audiovisual stimulus, but not with the food.
- The nausea (toxin or the x-ray) with food, but not with the audiovisual stimulus.

Namely, when reacting to nausea, rats seem to prefer hypotheses related to food, and to avoid hypotheses related to audiovisual stimulus. This is an inductive bias. Similarly, when reacting to physical trauma (electric shock), rats seem to prefer hypotheses related to audiovisual stimulus, and disfavor hypotheses related to food.

Notice that in Corollary 11 we saw that even if we know exactly what the unknown distribution $\mathcal{D}$ is (specifically, if we know that it is the uniform distribution over $\mathcal{X}$), we still cannot generalize well. Therefore, a reasonable approach to inductive bias is to make assumptions on the target function f. In particular, a natural assumption which we will explore in depth, is that f belongs to some subset, or *hypothesis class*, $\mathcal{H} \subseteq \mathcal{F}$, where $\mathcal{F}$ is the set of all functions from $\mathcal{X} \to \mathcal{Y}$. We will see that if $\mathcal{H}$ is simple in some sense, or has some structure, then generalization is indeed possible.

Formally, adopting the assumption that $f \in \mathcal{H}$ for some class $\mathcal{H} \subseteq \mathcal{F}$ yields the following definition of learning.

▶ **Definition 13** (PAC Learning, Valiant [5]). *Let $\mathcal{X}$ and $\mathcal{Y}$ be nonempty sets, let $\mathcal{F}$ be the set of all functions from $\mathcal{X} \to \mathcal{Y}$, and let $\mathcal{H} \subseteq \mathcal{F}$ be a class of functions. We say that a (possibly randomized) algorithm A is a probably approximately correct (PAC) learner for $\mathcal{H}$ if there exists a sample complexity function $m : (0,1)^2 \to \mathbb{N}$ such that for every precision parameter $\varepsilon \in (0,1)$, every confidence parameter $\delta \in (0,1)$, every target function $f \in \mathcal{H}$, and every distribution $\mathcal{D}$ over $\mathcal{X}$, if A receives as input the parameters ε and δ and a sample S of size $m = m(\varepsilon, \delta)$ such that $S = \Big((x_1, f(x_1)), \ldots, (x_m, f(x_m))\Big)$ where $x_1, \ldots, x_m$ are sampled independently from $\mathcal{D}$, then A halts and outputs a hypothesis $h \in \mathcal{F}$ that with probability at least $1 - \delta$ (over the sample S and the randomness of A) has loss $L_{\mathcal{D},f}(h) \leq \varepsilon$.*

We say that a class $\mathcal{H} \subseteq \mathcal{F}$ is PAC learnable if there exists an algorithm that is a PAC learner for $\mathcal{H}$.

This definition is similar to Definition 2, the difference being that we assume that $f \in \mathcal{H}$. This is called the *realizability* assumption. (We have also explicitly included the sample complexity function $m(\varepsilon, \delta)$ instead of writing "$\forall\varepsilon\,\forall\delta\,\exists m$," but that is purely a stylistic issue.)

1.5 Finite Hypothesis Classes are PAC Learnable

Perhaps the simplest assumption to make about the hypothesis class $\mathcal{H}$ is that it is finite. We now show that this suffices to ensure PAC learnability, even if the domain is infinite.

▶ **Lemma 14.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be (finite or infinite) nonempty sets, and let $\mathcal{H}$ be a finite subset of the functions $\mathcal{X} \to \mathcal{Y}$. Then $\mathcal{H}$ is PAC learnable with sample complexity $m(\varepsilon, \delta) = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon} \right\rceil$.*

The idea is similar to that of Claim 5, which showed that naïve PAC learning is possible for finite domains using the memorization algorithm. However, observe that the memorization algorithm does *not* satisfy Lemma 14. Intuitively, in order to avoid the no-free-lunch theorems, we need an algorithm which utilizes inductive bias, namely, which uses the assumption that $f \in \mathcal{H}$ to its advantage – but memorization doesn't do this. More concretely, Corollary 11 states that for any algorithm A (including a memorization algorithm), there exists a target function $f_{\text{fail-}A}$ for which the algorithm A fails, and so in particular A is a not a PAC learner for the finite class $\mathcal{H} = \{f_{\text{fail-}A}\}$. To avoid this issue, we need to design our algorithm A in a manner that depends on the class $\mathcal{H}$ so that $f_{\text{fail-}A} \notin \mathcal{H}$. Thus, under the assumption that the target function $f \in \mathcal{H}$, we know that A will not be executed with the target function $f = f_{\text{fail-}A}$, and so it is possible for A to succeed.

How can we design a learner A that uses the assumption that $f \in \mathcal{H}$, and in particular ensures that $f_{\text{fail-}A} \notin \mathcal{H}$? An approach called *empirical risk minimization (ERM)* does the trick.

$\text{ERM}_{\mathcal{H}}(S)$:
  find $h \in \mathcal{H}$ such that $L_S(h) = 0$
  output h

Algorithm 2 The empirical risk minimization algorithm. We assume that $\mathcal{H}$ is a class of functions and S is a sample. From the realizability assumption, there exists at least one hypothesis $h \in \mathcal{H}$ such that $L_S(h) = 0$. If there are multiple such hypotheses, the algorithm chooses one of them arbitrarily.

We can think of $\text{ERM}_{\mathcal{H}}$ as a constrained version of the memorization algorithm. It still outputs a hypothesis h such that $h(x_i) = y_i$ for all x_i in the sample, but in addition it satisfies the constraint that $h \in \mathcal{H}$.

The proof is very similar to that of Claim 5.

Proof of Lemma 14. Fix a finite class $\mathcal{H}$, a target function $f \in \mathcal{H}$, parameters $\varepsilon > 0$ and $\delta > 0$, and a distribution D. We show that

$$\mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[L_{\mathcal{D},f}(\text{ERM}_{\mathcal{H}}(S)) > \varepsilon\right] < \delta$$

for $m = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon} \right\rceil$.

Let $h \in \mathcal{H}$. We say that the sample S is *bad for* h if $L_S(h) = 0$ and $L_{\mathcal{D}}(h) > \varepsilon$. Notice that if $L_{\mathcal{D}}(\mathrm{ERM}_{\mathcal{H}}(S)) > \varepsilon$, then S is bad for the hypothesis $h = \mathrm{ERM}_{\mathcal{H}}(S)$. Hence,

$$\begin{aligned}
\mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[L_{\mathcal{D},f}(\mathrm{ERM}_{\mathcal{H}}(S)) > \varepsilon\right] &\leq \mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[\exists h \in \mathcal{H} : S \text{ is bad for } h\right] \\
&= \mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[\bigcup_{h\in\mathcal{H}} \{S \text{ is bad for } h\}\right] \\
&\leq \sum_{h\in\mathcal{H}} \mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[S \text{ is bad for } h\right]. \qquad \text{(union bound)}
\end{aligned}$$

We now show that $\forall h \in \mathcal{H} :\ \mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[S \text{ is bad for } h\right] < \delta/|\mathcal{H}|$. Fix $h \in \mathcal{H}$. If $L_{\mathcal{D}}(h) \leq \varepsilon$ then S is never bad for h. Otherwise,

$$\begin{aligned}
\mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[S \text{ is bad for } h\right] &= \mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[L_S(h) = 0\right] \\
&= \mathbb{P}_{S=((x_1,y_1),\ldots,(x_m,y_m))\sim(\mathcal{D},f)^m}\left[\bigcap_{i\in[m]} \{h(x_i) = f(x_i)\}\right] \\
&= \prod_{i\in[m]} \mathbb{P}_{S\sim(\mathcal{D},f)^m}\left[h(x_i) = f(x_i)\right] \qquad (x_i\text{'s are i.i.d.}) \\
&< (1-\varepsilon)^m \qquad (L_{\mathcal{D}}(h) > \varepsilon) \\
&\leq e^{-\varepsilon m} \leq \delta/|\mathcal{H}|,
\end{aligned}$$

where the last inequality holds whenever $m \geq \ln(|\mathcal{H}|/\delta)/\varepsilon$. By combining the two chains of inequalities above, we see that taking a sample of size $m(\varepsilon, \delta)$ as in the statement suffices to ensure that $\mathbb{P}\left[L_{\mathcal{D},f}(\mathrm{ERM}_{\mathcal{H}}(S)) > \varepsilon\right] < \sum_{h\in\mathcal{H}} \delta/|\mathcal{H}| = \delta$, as desired. ◀

2 Discussion

So far we have seen two extreme cases: if the hypothesis class $\mathcal{H}$ is finite, then it is PAC learnable, while if $\mathcal{H}$ is the set of all functions over an infinite domain, then it is not PAC learnable. But there are many useful classes that lie in between these two extremes. A major goal for us will be to chart this territory. Ideally, we would like to find a simple characterization that applies to all classes, and tells us precisely which classes are PAC learnable, and with what sample complexity.

Additionally, our definition of PAC learning required a relatively strong assumption, namely that the labels of the unknown system perfectly match some function f. But in many real-world scenarios this is not the case. Next lecture we will see how we can modify our definition of learnability to model these scenarios as well.

2.1 Bibliographic Notes

Good introductions to PAC learning are available in [4, Chapters 3-5] and [3, Chapter 2]. A good introduction to Hume's problem of induction is available in [2].

References

1 John Garcia and Robert A. Koelling. Relation of cue to consequence in avoidance learning. *Psychonomic science*, 4(1):123–124, 1966.

2 Leah Henderson. The Problem of Induction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2020 edition, 2020.

3 Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2nd edition, 2018.

4 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning – From Theory to Algorithms*. Cambridge University Press, 2014. URL: `http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms`.

5 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. `doi:10.1145/1968.1972`.

6 David H. Wolpert. The Lack of A-Priori Distinctions Between Learning Algorithms. *Neural Comput.*, 8(7):1341–1390, 1996. `doi:10.1162/neco.1996.8.7.1341`.