CPSC 532D — 7. PAC LEARNING; NO FREE LUNCH

Danica J. Sutherland

University of British Columbia, Vancouver

Fall 2025

7.1 PAC LEARNING

Our VC bound of Theorem 6.11 – and indeed our finite-class bound of Proposition 2.2 – have one striking property, which is notably *not* shared by our bound on logistic regression from Equation (5.9): they give bounds on the excess error of ERM which don't depend on *anything* about the distribution $\mathcal{D}$, only $\mathcal{H}$, ℓ, and m.

This property corresponds to one of the standard notions of learnability. Here, we're going to use a general idea of a learning algorithm as some function that takes a sample $S \in \mathcal{Z}^*$ (the set of sequences of any length from $\mathcal{Z}$) and returns a hypothesis in $\mathcal{H}$.

Definition 7.1. An algorithm $\mathcal{A} : \mathcal{Z}^* \to \mathcal{H}$ *agnostically PAC learns* $\mathcal{H}$ with a loss ℓ if there exists a function $m : (0,1)^2 \to \mathbb{N}$ such that, for every $\varepsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{Z}$, for any $m \geq m(\varepsilon, \delta)$, we have that

$$\Pr_{S \sim \mathcal{D}^m, \mathcal{A}}\left(L_{\mathcal{D}}(\mathcal{A}(S)) > \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon \right) < \delta,$$

where the randomness is both over the choice of S and any internal randomness in the algorithm $\mathcal{A}$. That is, $\mathcal{A}$ can *probably* get an *approximately correct* answer, where "correct" means the best possible loss in $\mathcal{H}$.

If $\mathcal{A}$ runs in time polynomial in $1/\varepsilon$, $1/\delta$, m, and some notion of the size of h^*, then we say that $\mathcal{A}$ *efficiently agnostically PAC learns* $\mathcal{H}$.

Definition 7.2. A hypothesis class $\mathcal{H}$ is *agnostically PAC learnable* if there exists an algorithm $\mathcal{A}$ which agnostically PAC learns $\mathcal{H}$.

So, ERM agnostically PAC-learns finite hypothesis classes for losses bounded in $[a, b]$, with the sample complexity $m(\varepsilon, \delta) = \frac{2(b-a)^2}{\varepsilon^2} \log \frac{|\mathcal{H}|+1}{\delta}$, from Proposition 2.2. More generally, Theorem 6.11 shows that ERM PAC-learns finite-VC classes for binary classification with zero-one loss, with the sample complexity being any function $m(\varepsilon, \delta)$ such that

A finite hypothesis class necessarily has finite VC dimension (why?).

$$\sqrt{\frac{2}{m(\varepsilon,\delta)}} \left[\sqrt{d(\log(m(\varepsilon,\delta)) + 1 - \log d)} + \sqrt{\log \frac{2}{\delta}} \right] \leq \varepsilon;$$

a little bit of annoying algebra can give us that it suffices to have, say, the sample complexity $m(\varepsilon, \delta) \geq \frac{8}{\varepsilon^2} \max\left\{ \log \frac{2}{\delta}, 4d \log \frac{32d}{\varepsilon^2} \right\}$.

In the definition of agnostic PAC learning, there's no limitation at all on the distribution – there needs to be an $m(\varepsilon, \delta)$ that works for *any* $\mathcal{D}$. This is nice, in that

For more, visit https://cs.ubc.ca/~dsuth/532D/25w1/.

since $\mathcal{H}$ and ℓ are under our control and m is easy to observe, we can be *sure* that it applies to whatever problem we're thinking about. It is, however, an extremely worst-case kind of notion; if we know more about $\mathcal{D}$, we might be able to get better bounds. For example, maybe the number of samples you need depends on "how hard" the particular problem is. We'll talk about this more a little later in the course ("nonuniform learning"). For now, it's worth mentioning one common special case:

Definition 7.3. Consider a nonnegative loss $\ell(h, z) \geq 0$. A distribution $\mathcal{D}$ is called *realizable* by $\mathcal{H}$ if there exists an $h^* \in \mathcal{H}$ such that $L_{\mathcal{D}}(h^*) = 0$.

This version doesn't have the "agnostic" modifier because it was introduced first [Val84].

Definition 7.4. An algorithm $\mathcal{A} : \mathcal{Z}^* \to \mathcal{H}$ is said to (realizably) *PAC learn* $\mathcal{H}$ with a loss ℓ if there exists a function $m : (0,1)^2 \to \mathbb{N}$ such that, for every $\varepsilon, \delta \in (0,1)$, for every *realizable* distribution $\mathcal{D}$ over $\mathcal{Z}$, for any $m \geq m(\varepsilon, \delta)$, we have that

$$\Pr_{S \sim \mathcal{D}^m, \mathcal{A}} (L_{\mathcal{D}}(\mathcal{A}(S)) > \varepsilon) < \delta,$$

where the randomness is both over the choice of S and any internal randomness in the algorithm $\mathcal{A}$. That is, $\mathcal{A}$ can *probably* get an *approximately correct* answer, where "correct" means zero loss.

If $\mathcal{A}$ runs in time polynomial in $1/\varepsilon$, $1/\delta$, m, and some notion of the size of h^*, then we say that A *efficiently (realizably) PAC learns* $\mathcal{H}$.

Definition 7.5. A hypothesis class $\mathcal{H}$ is *PAC learnable* if there exists an algorithm $\mathcal{A}$ which PAC learns $\mathcal{H}$.

Sometimes people say "realizable PAC learnable" or similar, to emphasize the difference versus agnostic PAC. The name "agnostic" is because the definition doesn't care whether there's a perfect h^* or not. (Notice that if $\mathcal{A}$ agnostically PAC learns $\mathcal{H}$, then it also PAC learns $\mathcal{H}$.)

The emphasis here on "how many samples for a given error" is also kind of a TCS-style framing, whereas statisticians more often ask "how much error for a given number of samples"; I tend to prefer the latter, but it's all equivalent.

If you read [SSBD14] or other work by computational learning theorists, there tends to be a lot of focus on just being learnable versus not being learnable. That problem has been solved (for binary classification), though, as we'll see soon! Recent work focuses much more on rates than on just learnability or not, and tends to be willing to make *some* assumptions on $\mathcal{D}$ rather than either being totally general or assuming only realizability.

A related question: maybe we know a better upper bound for algorithm A than algorithm B; does that mean A is actually better than B?

We've shown that anything with finite VC dimension is PAC learnable, and that ERM does so. But that's just an upper bound, and maybe we just didn't prove a good enough bound. Maybe it could turn out that ERM also PAC learns some infinite VC classes, or maybe it doesn't but some other algorithm does. As we'll see next, neither situation is true.

7.2 NO FREE LUNCH FOR HIGH-VC CLASSES

This result is similar to Theorem 5.1 of [SSBD14], but incorporating the idea of VC dimension (which they haven't introduced yet at that point).

Theorem 7.6. *Let $\mathcal{H}$ be a hypothesis set of binary classifiers over $\mathcal{X}$. Let $m \leq \operatorname{VCdim}(\mathcal{H})/2$. Then, using 0-1 loss,*

$$\inf_{\mathcal{A}} \sup_{\mathcal{D} \text{ realizable by } \mathcal{H}} \Pr_{\substack{S \sim \mathcal{D}^m \\ \mathcal{A}}} \left(L_{\mathcal{D}}(\mathcal{A}(S)) > \frac{1}{8} \right) \geq \frac{1}{7},$$

where the infimum over $\mathcal{A}$ is over all (possibly randomized) learning algorithms which return hypotheses in $\mathcal{H}$, and the probability is over both the sampling of a training set *and any internal randomness in $\mathcal{A}$.*

Before we prove this, let's unpack the quantifiers a bit. For any m and any learning algorithm $\mathcal{A}$, there is some realizable distribution $\mathcal{D}$ such that $\mathcal{A}$ has at least constant probability of failing with m samples, i.e. getting at least 1/8 error. Note that this distribution *depends on* m and on $\mathcal{A}$.

This result immediately implies the following:

Corollary 7.7. *Any* $\mathcal{H}$ *with* $\operatorname{VCdim}(\mathcal{H}) = \infty$ *is not PAC learnable.*

This doesn't necessarily mean that there's any single $\mathcal{D}$ that $\mathcal{A}$ fails on forever. But, at any m, there's still *some distribution* that's too hard. This removes the possibility of PAC learning, which needs to work for *all* distributions at a uniform rate.

Proof of Theorem 7.6. We're first going to pick a shatterable set of size $2m$, $\tilde{\mathcal{X}} = \{\tilde{x}_1, \dots, \tilde{x}_{2m}\} \subseteq \mathcal{X}$; at least one such set must exist, since $2m \le \operatorname{VCdim}(\mathcal{H})$. Then we'll pick the marginal distribution of x, $\mathcal{D}_x$, to be a discrete uniform distribution on $\tilde{\mathcal{X}}$. To construct our hard $\mathcal{D}$, we're going to use this $\mathcal{D}_x$ and then somehow assign a y for each x.

Since we're being totally generic with respect to $\mathcal{A}$, it's going to be hard to say which $y \mid x$ labeling rule in particular is going to be hard for $\mathcal{A}$ to learn. So, as a proof technique, we're going to start with a *random* labeling rule, and then settle on a particular one later. Specifically, for each vector of possible labels $y \in \{0,1\}^m$, choose some particular $f \in \mathcal{H}$ such that $f(x_j) = y_j$ for all j; there must be at least one, since $\mathcal{H}$ shatters $\tilde{\mathcal{X}}$. Let $\mathcal{F}$ be the set of these functions (of size exactly 2^m), and choose $f \sim \operatorname{Unif}(\mathcal{F})$, i.e. we're picking a labeling function uniformly from $\mathcal{F}$. For any f, let the distribution $\mathcal{D}_{(f)}$ denote the distribution that you get by sampling $x \sim \mathcal{D}_x$ and then assigning $y \mid x = f(x)$.

Now, for any sample of inputs $S_x = (x_1, \dots, x_m)$, we can implicitly construct a sample of pairs $S = \big((x_1, f(x_1)), \dots, (x_m, f(x_m))\big)$. Run the algorithm $\mathcal{A}$ to get $\hat{h}_S = \mathcal{A}(S)$, which itself might be random given S. Its expected loss over the process of choosing a distribution, sampling a training set, and running the algorithm is

$$\mathop{\mathbb{E}}_{f \sim \operatorname{Unif}(\mathcal{F})} \mathop{\mathbb{E}}_{S \sim \mathcal{D}_{(f)}^m} \mathop{\mathbb{E}}_{\mathcal{A}} L_{\mathcal{D}_{(f)}}(\mathcal{A}(S)) = \mathop{\mathbb{E}}_{f} \mathop{\mathbb{E}}_{S} \mathop{\mathbb{E}}_{\mathcal{A}} \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} \mathbb{1}\big([\mathcal{A}(S)](x) \ne f(x)\big).$$

Using the law of total expectation, let's break this expectation up based on whether the test x is in the training data S or not:

$$\begin{aligned}\mathop{\mathbb{E}}_{f,S,\mathcal{A}} \mathop{\mathbb{E}}_{x} \mathbb{1}([\mathcal{A}(S)](x) \ne f(x)) = \mathop{\mathbb{E}}_{f,S,\mathcal{A}} \Big[&\Pr(x \in S_x) \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} [\mathbb{1}([\mathcal{A}(S)](x) \ne f(x)) \mid x \in S_x] \\ &+ \Pr(x \notin S_x) \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} [\mathbb{1}([\mathcal{A}(S)](x) \ne f(x)) \mid x \notin S_x] \Big].\end{aligned}$$

For the first term, we're not going to worry about what the algorithm does on the data it's actually seen, since the algorithm might be good: we'll just bound this as being at least zero. For the second, we know since $\mathcal{D}_x$ is uniform and $|S_x| \le m$ that

$$\Pr(x \notin S_x) = \frac{|\tilde{\mathcal{X}} \setminus S_x|}{|\tilde{\mathcal{X}}|} \ge \frac{m}{2m} = \frac{1}{2}.$$

Thus we've shown that

$$\mathop{\mathbb{E}}_{f\sim\mathrm{Unif}(\mathcal{F})}\ \mathop{\mathbb{E}}_{S\sim\mathcal{D}_{(f)}^m}\ \mathop{\mathbb{E}}_{\mathcal{A}} L_{\mathcal{D}_{(f)}}(\mathcal{A}(S)) \ge \frac{1}{2} \mathop{\mathbb{E}}_{f,S,\mathcal{A}}\ \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}[\mathbb{1}([\mathcal{A}(S)](x) \ne f(x)) \mid x \notin S_x].$$

Now, I'll show this remaining expectation is exactly 1/2, since our labels $f(\tilde{x}_j)$ are uniformly random and totally independent of the training set. To see this more clearly, let's change notation a little. Rather than choosing an f and then an $S \sim \mathcal{D}_{(f)}^m$, let's think of choosing the x points S_x, and the labels for *every* point in $\tilde{X}$. Once we know both of those things, we can easily construct S. Let $I \sim \mathrm{Unif}([2m])^m$ be the sequence of random indices into $\tilde{X}$, and let $\tilde{y} \sim \mathrm{Unif}(\pm 1)^{2m}$ be the vector of labels for every possible input. Then $S = \Big((\tilde{x}_{I_1}, \tilde{y}_{I_1}), \dots, (\tilde{x}_{I_m}, \tilde{y}_{I_m})\Big)$, and

$$\begin{aligned}
&\mathop{\mathbb{E}}_{f\sim\mathrm{Unif}(\mathcal{F})}\ \mathop{\mathbb{E}}_{S\sim\mathcal{D}_{(f)}^m}\ \mathop{\mathbb{E}}_{\mathcal{A}} L_{\mathcal{D}_{(f)}}(\mathcal{A}(S)) \\
&\qquad \ge \frac{1}{2} \mathop{\mathbb{E}}_{I\sim\mathrm{Unif}([2m])^m}\ \mathop{\mathbb{E}}_{\tilde{y}\sim\mathrm{Unif}(\pm1)^{2m}}\ \mathop{\mathbb{E}}_{\mathcal{A}}\ \mathop{\mathbb{E}}_{j\sim\mathrm{Unif}([2m]\setminus I)}[\mathbb{1}([\mathcal{A}(S)])(\tilde{x}_j) \ne \tilde{y}_j)] \\
&\qquad = \frac{1}{2} \mathop{\mathbb{E}}_{I\sim\mathrm{Unif}([2m])^m}\ \mathop{\mathbb{E}}_{\tilde{y}_I\sim\mathrm{Unif}(\pm1)^{|I|}}\ \mathop{\mathbb{E}}_{\mathcal{A}}\ \mathop{\mathbb{E}}_{j\sim\mathrm{Unif}([2m]\setminus I)}\ \mathop{\Pr}_{\tilde{y}_j\sim\mathrm{Unif}(\pm1)}\Big([\mathcal{A}(S)](\tilde{x}_j) \ne \tilde{y}_j\Big).
\end{aligned}$$

Whatever the choice of $[\mathcal{A}(S)](\tilde{x}_j)$, it has absolutely nothing to do with $\tilde{y}_j$; thus whether our prediction agrees with the label is just a pure coin flip, 50% chance it agrees, 50% it doesn't. That is,

$$\mathop{\mathbb{E}}_{f\sim\mathrm{Unif}(\mathcal{F})}\ \mathop{\mathbb{E}}_{S\sim\mathcal{D}_{(f)}^m}\ \mathop{\mathbb{E}}_{\mathcal{A}} L_{\mathcal{D}_{(f)}}(\mathcal{A}(S)) \ge \tfrac{1}{4}.$$

This proof technique is known as the probabilistic method, *and often attributed to Paul Erdős.*

But, if the *average* over f of the expected loss $\mathbb{E}_{S\sim\mathcal{D}_{(f)}^m} L_{\mathcal{D}_{(f)}}(\hat{h}_S)$ is at least $\frac{1}{4}$, then there must be at least one *particular* f such that the expected loss is at least $\frac{1}{4}$! Pick one and call it g; this will be the labeling function claimed by the theorem.

We've now shown the average loss is large, but we still want to show that the loss has high probability of being large. Now, $L_{\mathcal{D}_{(g)}}(\mathcal{A}(S))$ is a random variable bounded in $[0, 1]$, and we already know one way to bound those variables in terms of their means: Markov's inequality. But Markov's inequality bounds the probability of things being *big*, and we want to bound the probability of this being *small*. So we'll need to switch it around, which is sometimes called "reverse Markov":

$$\Pr(L_{\mathcal{D}_{(g)}}(\hat{h}_S) \le \tfrac{1}{8}) = \Pr\left(1 - L_{\mathcal{D}_{(g)}} \ge 1 - \frac{1}{8}\right) \le \frac{1 - \mathbb{E} L_{\mathcal{D}_{(g)}}(\hat{h}_S)}{\frac{7}{8}} \le \left(1 - \frac{1}{4}\right)\frac{8}{7} = \frac{6}{7}.$$

Thus, for the realizable $\mathcal{D}_{(g)}$ we picked above,

$$\mathop{\Pr}_{\mathcal{A};S\sim\mathcal{D}_{(g)}^m}\left(L_{\mathcal{D}_{(g)}}(\mathcal{A}(S)) > \frac{1}{8}\right) \ge \frac{1}{7}. \qquad \square$$

7.2.1 *Interpretation*

Theorem 7.6 is sometimes called a "no free lunch" theorem, in that there is no algorithm that *always* works (in the sense of PAC learning): every algorithm fails on at least one distribution.

In fact, basically this same proof strategy implies [Wol96] that, if you only care about the "off-sample" error (the average error on $(x, y) \mid x \notin S_x$), there are just as many possible distributions where your predictor is right as where it's wrong, regardless of your learning algorithm. If you don't assume *anything* about the world, all algorithms perform the same on average over all possible worlds.

This is in some ways a deep philosophical problem, called the problem of induction and generally credited to David Hume. The fact that the sun rose every day so far doesn't, from "pure first principles," imply anything about whether it will rise tomorrow: we just decide to prefer "simple" explanations, i.e. we choose some $\mathcal{H}$ that we like. But that doesn't really answer which $\mathcal{H}$ would be good.

Actually, VC or Rademacher theory can't answer that problem either: it's preferable to choose a $\mathcal{H}$ with small complexity, but since $\operatorname{Rad}((\mathcal{H} + \{f\})|_S) = \operatorname{Rad}(\mathcal{H}|_S)$, and $\operatorname{VCdim}(\mathcal{H}) = \operatorname{VCdim}(\{x \mapsto h(x)f(x) : h \in \mathcal{H}\})$ for ± 1-valued h and f, we haven't actually seen any objective notion of a "simple hypothesis"! We've only seen ways to say that *sets* of hypotheses are all similar enough to one another.

Sometimes people get a little mystical about no free lunch theorems, though – e.g. `no-free-lunch.org` says that this result "calls the whole of science into question," and I've heard a story about a certain famous machine learning theorist who, when asked to help scientists on some applied problem, supposedly said something to the effect of "because of the no free lunch theorem, since I don't know anything about your field then nothing I do can possibly help."

But the world is *not* uniformly random; we know from experience that some kinds of $\mathcal{H}$ tend to work better than others. so, although there is *some* distribution that every algorithm fails on, it's not the case in the world we live in that all algorithms are the same as each other. (And, interestingly, there *are* (impractical) learning algorithms that are always at least as good as any other algorithm, up to (huge) constants: check out `free-lunch.org` [Nak21].)

7.2.2 *Aside: "learning is NP-hard"*

Another example of this kind of claim (based on a different underlying theorem) is given by van Rooij et al. [vRoo+24], who say (in reaction to recent progress of LLMs):

> [We present] a mathematical proof of inherent intractability (formally, NP-hardness) of the task that [...] AI engineers set themselves. This intractability implies that any factual AI system created in the short-run (say, within the next few decades or so) is so astronomically unlikely to be anything like a human mind, or even a coherent capacity that is part of that mind, that claims of 'inevitability' of AGI within the foreseeable future are revealed to be false and misleading. We realize that this implication may appear counterintuitive given everyday experiences and interactions with currently impressive AI systems, but we will explain why it is not. As we will carefully unpack later in the paper, it is a mistake to assume that AI systems' performance is either currently human-level, or will simply continue to improve and the systems will soon constitute human-level A(G)I. The problem is that—in line with our intractability result—the performance cannot scale up.

What they actually prove (their Theorem 2) can be rephrased roughly as follows:

Theorem 7.8 ("Ingenia Theorem", [vRoo+24]). *Let $\mathcal{X} = \{0,1\}^N$ and $\mathcal{Y}$ a fixed finite set. Let $\mathcal{H}$ be a hypothesis class containing all functions implemented by circuits with*

The paper actually allows a set of possible behaviours in response to a given input. Unfortunately, the formal statement in the paper doesn't quite make sense. I think, *from reading the* *complexity at most a parameter* D*; for instance, for each* N *and* D *there exists a class of feedforward neural networks satisfying this. Suppose that there exists a polynomial-time algorithm, allowed to randomly sample from* $\mathcal{D}$ *as a constant-time operation, which with probability at least* $\Omega(1/N^\alpha)$ *for some* $\alpha > 0$ *under any realizable* $\mathcal{D}$ *successfully identifies a hypothesis* $h \in \mathcal{H}$ *satisfying that*

$$\Pr_{(x,y)\sim\mathcal{D}}(h(x) = y) \geq \frac{1}{|\mathcal{Y}|} + \varepsilon_N,$$

for some $\varepsilon_N = \Omega(1/N^\beta)$*, for some* $\beta > 0$*. Then* NP $\subseteq$ BPP*.*

The assumption on the algorithm here is much weaker than efficient PAC learning: it's just that the learning algorithm does nontrivially better than random guessing, with nontrivial probability. But the conclusion NP $\subseteq$ BPP contradicts a *very* common assumption in complexity theory. So, although we don't 100% know this for a fact, we should probably think that this implies there is no polynomial-time algorithm that can improve on random guessing for any realizable distribution.

Does this imply that "AI" is computationally infeasible? **Not really.** Assuming NP $\not\subseteq$ BPP, it implies that for any given polynomial-time learning algorithm, there exist *some* distributions which cannot be efficiently learned. (This is true even for distributions which are themselves efficiently computable. The universal induction approach considered e.g. by Nakkiran [Nak21] finds computationally-efficient hypotheses but it does so in an extremely computationally-inefficient way.)

This obviously doesn't mean, though, that *every* distribution can't be efficiently learned. For instance, the distribution that always says "banana please" in response to any input at all can be. Is "human-like behaviour" a distribution that can be efficiently learned by some algorithm? I don't know (other than to say that, well, humans do it), and this theorem doesn't say either!

7.3 LOWER BOUNDS

Theorem 7.6 only applies when $m \leq \mathrm{VCdim}(\mathcal{H})/2$. We can use it, though, to also get a quantitative lower bound for higher m:

This theorem roughly follows [MRT18, Theorem 3.20]. That result merges this result with Theorem 7.6 in a way I find really hard *to follow; their theorem statement is also obviously incorrect when* $m < (\mathrm{VCdim}(\mathcal{H}) - 1)/32$*. [SSBD14, Theorem 6.8] states a similar result, but leaves this part as an exercise.*

Theorem 7.9. *Let* $\mathcal{H}$ *be a set of binary classifiers over* $\mathcal{X}$ *such that* $\mathrm{VCdim}(\mathcal{H}) \geq 2$*. For any* $m > \mathrm{VCdim}(\mathcal{H})/2$*,*

$$\inf_{\mathcal{A}} \sup_{\mathcal{D} \text{ realizable by } \mathcal{H}} \Pr_{S\sim\mathcal{D}^m}\left(L_{\mathcal{D}}(\mathcal{A}(S)) > \frac{\mathrm{VCdim}(\mathcal{H}) - 1}{32m}\right) > \frac{1}{100}$$

where $L_{\mathcal{D}}$ *uses zero-one loss, and the infimum over* $\mathcal{A}$ *is over all learning algorithms returning hypotheses in* $\mathcal{H}$*.*

Proof. Choose a set $\tilde{\mathcal{X}} = \{\tilde{x}_1, \ldots, \tilde{x}_d\}$ of size $d = \mathrm{VCdim}(\mathcal{H})$ which can be shattered by $\mathcal{H}$. We're going to choose a distribution that puts most of its probability mass on $\tilde{x}_1$, in such a way that we're likely to see fewer than half of the *other* points from the distribution. Specifically, for an $\varepsilon > 0$ to choose later,

$$\Pr_{x\sim\mathcal{D}_x}(x = \tilde{x}_1) = 1 - \varepsilon, \qquad \text{for all } i > 1, \ \Pr_{x\sim\mathcal{D}_x}(x = \tilde{x}_i) = \frac{\varepsilon}{d-1}.$$

Now, let $\tilde{\mathcal{D}}$ be the distribution over $\{\tilde{x}_2, \ldots, \tilde{x}_d\}$ selected by Theorem 7.6 with $m = (d-1)/2$, and let $f \in \mathcal{H}$ be the labeling function chosen in $\tilde{\mathcal{D}}$. Our distribution will be found by sampling $x \sim \mathcal{D}_x$ and then letting $y \mid x = f(x)$.

Now, we're going to prove that it's fairly likely that samples from $\mathcal{D}_x$ contain at most $(d-1)/2$ of the non-$\tilde{x}_1$ points. How many points we *don't* see is a little annoying to characterize exactly, but we can get a bound based on

$$Q = \sum_{i=1}^{m} \mathbb{1}(x_i \neq \tilde{x}_1);$$

if we repeat any of the non-$\tilde{x}_1$ points, Q will double-count them, but it's a valid upper bound on the number of non-$\tilde{x}_1$ points we see. Notice that $\Pr(x_i \neq \tilde{x}_1) = \varepsilon$, and each of the indicators is iid Bernoulli(ε), so $Q \sim \text{Binomial}(m, \varepsilon)$.

A standard tail bound for binomial variables, Proposition 7.10 with $\gamma = 1$, shows that

$$\Pr(Q \geq 2m\varepsilon) \leq \exp\left(-\frac{1}{3} m\varepsilon\right).$$

To use this result, we want $2m\varepsilon = \frac{1}{2}(d-1)$; so, pick $\varepsilon = (d-1)/(4m)$. This is valid, since $m > d/2$ implies that $\varepsilon = \frac{d-1}{4m} < \frac{d-1}{2d} < \frac{1}{2}$. Then we see less than half of the non-$\tilde{x}_1$ points with probability at least

$$1 - \exp\left(-\frac{m}{3} \cdot \frac{d-1}{4m}\right) = 1 - \exp\left(-\frac{d-1}{12}\right) \geq 1 - \exp\left(-\frac{1}{12}\right) > 0.07,$$

since $1 - \exp(-1/12) \approx 0.07995$.

So, with more than 7% probability, a sample of size m from $\mathcal{D}$ will contain at most $(d-1)/2$ of the non-$\tilde{x}_1$ points. Then, Theorem 7.6 tells us that with probability at least 1/7, $L_{\tilde{\mathcal{D}}}(\mathcal{A}(S)) \geq \frac{1}{8}$. If this happens, this implies that $L_{\mathcal{D}}(\mathcal{A}(S)) \geq \frac{1}{8}\varepsilon = \frac{d-1}{32m}$, since the total probability of the non-$\tilde{x}_1$ points is exactly ε. So, we have more than a $\frac{1}{7} \cdot 7\% = 1\%$ chance of seeing $\frac{d-1}{32m}$ error on $\mathcal{D}$, as desired. □

Proposition 7.10. *If* $X \sim \text{Binomial}(m, p)$, *then for any* $\gamma > 0$ *it holds that*

$$\Pr(X \geq (1+\gamma)mp) \leq \exp\left(-\frac{1}{3} mp\gamma^2\right).$$

which is e.g. Theorem D.4 of [MRT18]. The proof technique is a bit different from how we proved Hoeffding/etc, and does not hold as generally; it's only for "sub-Bernoulli" (bounded) variables, not subgaussians. If you're curious, you should be able to follow their proof (which uses their Theorem D.3) just fine.

Agnostic case You can get a bigger error if you don't require $\mathcal{D}$ to be realizable: Theorem 3.23 of [MRT18] gives that for any m and $\mathcal{H}$,

$$\inf_{\mathcal{A}} \sup_{\mathcal{D}} \Pr\left(L_{\mathcal{D}}(\mathcal{A}(S)) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \geq \sqrt{\frac{d}{320m}}\right) \geq \frac{1}{64}. \tag{7.1}$$

Section 28.2 of [SSBD14] is similar. The proof strategy is like before, but we further make the rarely-seen points have random labels with probabilities close enough to 1/2 that they take a lot of samples to identify, but far enough from 1/2 that you do suffer significant loss if you get them wrong.

More generally These styles of theorems are sometimes called "minimax bounds," and algorithms are called "minimax-optimal" or simply "minimax" if they achieve the lower bound (usually only up to constants, though that's also sometimes called "rate-optimal"). In Chapter 6 we showed that ERM gets error $\widetilde{\mathcal{O}}_p(\sqrt{d/m})$, which combined with the agnostic result above shows that ERM is (up to log factors) rate-optimal for finite-VC classes. Although we haven't shown this [see SSBD14, Section 28.3; Zhang23, Section 6.5], ERM for binary classifiers achieves $\widetilde{\mathcal{O}}_p(d/m)$ error in the realizable setting, so by Theorem 7.9 ERM is also (up to log factors) minimax rate-optimal for realizable distributions too.

Minimax rates are also available for various other problems, including things like linear regression, density estimation, and optimization. We won't talk a lot about lower bounds in this course, but they can be really nice to know whether your learning algorithm is "good" or not. (The problem, though, is they tend to be extremely "worst-case," and might not be too informative about problems you're likely to actually see – similar to no free lunch arguments.)

7.4 THE "FUNDAMENTAL THEOREM OF STATISTICAL LEARNING"

We've now shown all the necessary parts for a pretty complete qualitative understanding of PAC learning for binary classifiers.

This name is only, as far as I know, used by [SSBD14].

Theorem 7.11 (Fundamental Theorem of Statistical Learning)**.** *For $\mathcal{H}$ a class of functions $h : \mathcal{X} \to \{0, 1\}$ and with the 0-1 loss, the following are equivalent:*

[SSBD14] use two-sided uniform convergence: in the setting of the theorem here, one-sided bounds imply two-sided ones, but (a) one-sided is what we really use, and (b) in more general settings the distinction can matter.

1. *Uniform convergence: for all $\varepsilon, \delta \in (0, 1)$, we have that $\sup_{h\in\mathcal{H}} L_\mathcal{D}(h) - L_S(h) < \varepsilon$ with probability at least $1 - \delta$ as long as $m \geq m^{UC}(\varepsilon, \delta) < \infty$.*
2. *Any ERM rule agnostically PAC-learns $\mathcal{H}$.*
3. *$\mathcal{H}$ is agnostically PAC learnable.*
4. *Any ERM rule PAC-learns $\mathcal{H}$.*
5. *$\mathcal{H}$ is PAC learnable.*
6. $\mathrm{VCdim}(\mathcal{H}) < \infty$.

Proof. 1 implying 2 is our usual argument:

$$L_\mathcal{D}(\hat{h}_S) \leq L_S(\hat{h}_S) + \sup_{h\in\mathcal{H}} L_\mathcal{D}(h) - L_S(h) \leq L_S(h^*) + \varepsilon \leq L_\mathcal{D}(h^*) + [L_S(h^*) - L_\mathcal{D}(h^*)] + \varepsilon,$$

plus Hoeffding on $L_S(h^*) - L_\mathcal{D}(h^*)$.

2 implying 3, and 4 implying 5, are immediate.

2 implying 4, and 3 implying 5, is also straightforward from the definitions.

Corollary 7.7 shows that 5 implies 6.

6 implying 1 is shown by Theorem 6.11. □

Theorem 6.8 of [SSBD14] gives a quantitative version, bounding the sample complexities in terms of the VC dimension, by collecting lower bounds like Theorem 7.9 and (7.1) and upper bounds like Theorem 6.11 and the realizable equivalent that we didn't prove.

REFERENCES

[MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talkwalkar. *Foundations of Machine Learning*. 2nd edition. MIT Press, 2018.

[Nak21] Preetum Nakkiran. *Turing-Universal Learners with Optimal Scaling Laws*. 2021. arXiv: 2111.05321.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[Val84] Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM* 27.11 (1984), pages 1134–1142.

[vRoo+24] Iris van Rooij, Olivia Guest, Federico Adolfi, Ronald de Haan, Antonina Kolokolova, and Patricia Rich. Reclaiming AI as a Theoretical Tool for Cognitive Science. *Computational Brain & Behavior* (2024).

[Wol96] David H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation* 8.7 (October 1996), pages 1341–1390.

[Zhang23] Tong Zhang. *Mathematical Analysis of Machine Learning Algorithms*. Pre-publication version. 2023.