CPSC 532D — 8. LOWER BOUNDS; NO FREE LUNCH

Danica J. Sutherland

University of British Columbia, Vancouver

Fall 2024

Armed with our knowledge of online learning from Chapter 7, let's now return to the offline setting from Chapters 1, 2 and 4 to 6.

Should this have just come before Chapter 7? Probably, but the teaching timing wasn't right. . . .

In the offline setting, so far we've only done upper bounds: in this case, we know we can learn at least this well. But if we only know upper bounds, we never really know how tight they are, and so we can never really know if one algorithm is better than another, or if a learner will really fail in some situation or if it's just that our proof wasn't good enough.

One way to approach this problem is with asymptotic results, as described e.g. by [Bach24] who summarizes and translates results from the classic textbook of van der Vaart [vdV98]. For instance, if $\mathcal{H} = \{h_w : w \in \mathcal{W}\}$ for some open set of possible parameters $\mathcal{W} \subseteq \mathbb{R}^D$, the loss is sufficiently "nice" as a function of w, and there's a minimizer $h^* = h_{w^*}$, then as long as some extra "niceness" assumptions also hold, it's true for the ERM that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m} L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) = \Theta\left(\frac{1}{m}\operatorname{Tr}\left(\left[\nabla_w^2 L_{\mathcal{D}}(h^*)\right]^{-1} \mathop{\mathbb{E}}_{z\sim\mathcal{D}}\left[(\nabla_w \ell(h_w, z))(\nabla_w \ell(h_w, z))^T|_{w=w^*}\right]\right)\right).$$

This gives a fast $1/m$ rate – better than the $1/\sqrt{m}$ we've gotten so far (except in A1 Q4) – and along the way it actually also tells us that $w - w^*$ is asymptotically Gaussian, and some other nice things. If we can evaluate the stuff inside the trace, we could also then explicitly say "this $\mathcal{H}$ converges faster than that one," or compare to an asymptotic rate for some different algorithm. But: the "niceness" assumptions don't always hold, the expressions aren't always easy to analyze, and they're purely asymptotic results, so we don't know whether they're a good approximation after $m = 20$ or only after $m = 100,000,000,000$.

Instead, let's use a different route to *lower bounds*, specifically focusing on binary classifiers where these things are easiest.

8.1 NO FREE LUNCH FOR HIGH-VC CLASSES

Theorem 8.1. *Let $\mathcal{H}$ be a hypothesis set of binary classifiers over $\mathcal{X}$. Let $m \le \mathrm{VCdim}(\mathcal{H})/2$. Then, using 0-1 loss,*

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).

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

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 $\mathcal{A}$.

For more, visit https://cs.ubc.ca/~dsuth/532D/24w1/.

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 8.2. *Any $\mathcal{H}$ with $\text{VCdim}(\mathcal{H}) = \infty$ 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 8.1. 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 $2m \leq \text{VCdim}(\mathcal{H})$. 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 $f \sim \text{Unif}(\mathcal{F})$, 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 $\mathrm{S}_x = (x_1, \dots, x_m)$, we can implicitly construct a sample of pairs $\mathrm{S} = \Big((x_1, f(x_1)), \dots, (x_m, f(x_m))\Big)$. Run the algorithm $\mathcal{A}$ to get $\hat{h}_{\mathrm{S}} = \mathcal{A}(\mathrm{S})$, 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

$$\mathop{\mathbb{E}}_{f \sim \text{Unif}(\mathcal{F})} \mathop{\mathbb{E}}_{\mathrm{S} \sim \mathcal{D}_{(f)}^m} \mathop{\mathbb{E}}_{\mathcal{A}} \mathrm{L}_{\mathcal{D}_{(f)}}(\mathcal{A}(\mathrm{S})) = \mathop{\mathbb{E}}_{f} \mathop{\mathbb{E}}_{\mathrm{S}} \mathop{\mathbb{E}}_{\mathcal{A}} \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} \mathbb{1}\Big([\mathcal{A}(\mathrm{S})](x) \neq f(x)\Big).$$

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:

$$\mathop{\mathbb{E}}_{f,\mathrm{S},\mathcal{A}} \mathop{\mathbb{E}}_{x} \mathbb{1}(\hat{h}_{\mathrm{S}}(x) \neq f(x)) = \mathop{\mathbb{E}}_{f,\mathrm{S},\mathcal{A}} \Bigg[\Pr(x \notin \mathrm{S}_x) \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} [\mathbb{1}(\hat{h}_{\mathrm{S}}(x) \neq f(x)) \mid x \notin \mathrm{S}_x] + \Pr(x \in \mathrm{S}_x) \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} [\mathbb{1}(\hat{h}_{\mathrm{S}}(x) \neq f(x)) \mid x \in \mathrm{S}_x] \Bigg].$$

For the second 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 first term, we know since $\mathcal{D}_x$ is uniform and $|\mathrm{S}_x| \leq m$ that

$$\Pr(x \notin \mathrm{S}_x) = \frac{\left|\tilde{\mathcal{X}} \setminus \mathrm{S}_x\right|}{\left|\tilde{\mathcal{X}}\right|} \geq \frac{m}{2m} = \frac{1}{2}.$$

Also, since our labels $f(\tilde{x}_j)$ are uniformly random and totally independent of one

another, and S is independent of those labels for points $\tilde{x} \notin S$, whether $\hat{h}_S$ agrees with f is just a pure coin flip: $\mathbb{E}_x[\mathbb{1}(\hat{h}_S(x) \neq f(x)) \mid x \notin S_x] = \frac{1}{2}$.

Combining, we know that

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

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.

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

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, $L_{\mathcal{D}_{(g)}}(\hat{h}_S)$ 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":

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

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

$$\Pr_{S \sim \mathcal{D}_{(g)}^m}\left(L_{\mathcal{D}_{(g)}}(\hat{h}_S) > \tfrac{1}{8}\right) \geq \frac{1}{7}.$$ □

8.1.1 *Interpretation*

Theorem 8.1 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 $\mathrm{Rad}((\mathcal{H} + \{f\})|_S) = \mathrm{Rad}(\mathcal{H}|_S)$, and $\mathrm{VCdim}(\mathcal{H}) = \mathrm{VCdim}(\{x \mapsto h(x)f(x) : h \in \mathcal{H}\})$ for ± 1-valued h and f, we haven't actually seen any objective notion of a "simple hypothesis": only 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. `https://no-free-lunch.org` says that this result "calls the whole of science into question." 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: `free-lunch.org` used to (but, alas, no longer) point to the paper of Nakkiran [Nak21].)

8.1.2 *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 8.3 ("Ingenia Theorem", [vRoo+24]). *Let $\mathcal{X} = \{0,1\}^N$ and $\mathcal{Y}$ a fixed finite set. For each x, define $\mathcal{Y}_x \subsetneq \mathcal{Y}$ to be the set of "acceptable" responses to an input x. Let $\mathcal{H}$ be a hypothesis class containing all functions implemented by circuits with complexity at most a parameter D; for instance, for each N and D there exists a class of feedforward neural networks satisfying this. A realizable distribution $\mathcal{D}$ is one where $\Pr_{(x,y)\sim\mathcal{D}}(y \in \mathcal{Y}_x) = 1$ and there exists $h^* \in \mathcal{H}$ with $\Pr_{(x,y)\sim\mathcal{D}}(h^*(x) \in \mathcal{Y}_x) = 1$. 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 $\alpha > 0$ successfully identifies a hypothesis $h \in \mathcal{H}$ satisfying*

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

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

This conclusion 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 satisfying the above properties, i.e. that can improve on random guessing.

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!

8.2 LOWER BOUNDS

Theorem 8.1 only applies when $m \le \text{VCdim}(\mathcal{H})/2$. We can use it, though, to also get a quantitative lower bound for higher m:

Theorem 8.4. *Let $\mathcal{H}$ be a set of binary classifiers over $\mathcal{X}$ such that $\text{VCdim}(\mathcal{H}) \ge 2$. For any $m > \text{VCdim}(\mathcal{H})/2$,*

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

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

This theorem roughly follows [MRT18, Theorem 3.20]. That result merges this result with Theorem 8.1 in a way I find really *hard to follow; their theorem statement is also obviously incorrect when $m < (\text{VCdim}(\mathcal{H})-1)/32$. [SSBD14, Theorem 6.8] states a similar result, but leaves this part as an exercise.*

Proof. Choose a set $\tilde{\mathcal{X}} = \{\tilde{x}_1, \dots, \tilde{x}_d\}$ of size $d = \text{VCdim}(\mathcal{H})$ 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 less 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 $\{\tilde{x}_2, \dots, \tilde{x}_d\}$ selected by Theorem 8.1 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

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

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 $\Pr(x_i \ne \tilde{x}_1) = \varepsilon$, and each of the indicators is iid Bernoulli(ε), so $Q \sim \text{Binomial}(m, \varepsilon)$.

A standard tail bound for binomial variables, Proposition 8.5 with $\gamma = 1$, shows that

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

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 $\varepsilon < \frac{1}{2}\frac{d-1}{d} < \frac{1}{2}$. Then we see less than half of the non-$\tilde{x}_1$ points with probability at least

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

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 8.1 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 8.5. *If* $X \sim \text{Binomial}(m, p)$, *then for any* $\gamma > 0$ *it holds that*

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

This is an immediate consequence of the multiplicative Chernoff bound, which is e.g. Theorem D.4 of [MRT18]. The proof technique is different from how we proved Hoeffding/etc, and I don't know if it holds as generally, but 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 $\mathcal{H}$,

$$\inf_{\mathcal{A}} \sup_{\mathcal{D}} \Pr\left(L_{\mathcal{D}}(\mathcal{A}(S)) - \inf_{h\in\mathcal{H}} L_{\mathcal{D}}(h) \geq \sqrt{\frac{d}{320m}}\right) \geq \frac{1}{64}. \tag{8.1}$$

Section 28.2 of [SSBD14] is similar.

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 the VC notes 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 Section 28.3 of [SSBD14] or 6.5 of [Zhang23]), ERM for binary classifiers achieves $\widetilde{\mathcal{O}}_p(d/m)$ error in the realizable setting, so by Theorem 8.4 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.)

8.3 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 8.6 (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:*

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.*

[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.

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:

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

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 8.2 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 8.4 and (8.1) and upper bounds like Theorem 6.11 and the realizable equivalent that we didn't prove.

REFERENCES

[Bach24] Francis Bach. *Learning Theory from First Principles*. Draft version. August 2024.

[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.

[vdV98] Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

[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.