Yale University S&DS 669
Statistical Learning Theory

Instructor: Dr. Omar Montasser
Scribe: Anish Lakkapragada

September 24, 2025

Contents

Chapter 1

PAC Learning and VC Theory

1.1 Course Logistics (Lecture 1)

We start the class by introducing our names & majors before getting into objectives of this course. Omar also covers the syllabus (recommended prerequisites, grading, and AI policy) before going over a roadmap of the things we will cover. Okay, let's start!

1.2 Introducing the Statistical Learning Theory Framework (Lecture 1)

We now introduce the statistical learning theory framework where we have the following objects:

- Domain X (e.g. $X = \mathbb{R}^d$) where each $x \in X$ is called an "instance"
- Label Space Y (e.g. $Y = \{\pm 1\}$ or $Y = \mathbb{R}$)
- Unknown source distribution D over $X \times Y$. This is an assumption on the data generating process (formed by "nature" or "reality").
- Goal: find a predictor $h : X \to Y$ achieving small *expected error* $L_D(h) := \mathbb{P}_{(x,y)\sim D}\{h(x) \neq y\}$.
- Access to an oracle: We have an i.i.d training sample $S = \{(x_i, y_i)\}_{i=1}^m$ drawn from D (notated by $S \sim D^m$)

Restated, our goal is to create some learner $A : (X \times Y)^\star \to Y^X$, where the $\star$ denotes a variable-length sequence of $X \times Y$ (i.e. our dataset) and Y^X is the set of all functions mapping from X to Y.

Omar notes that we will first start by assuming that any instance $x \in X$ has a "ground-truth" label, as opposed to a case where D allows for 50% probability mass on $(x, +1)$ and $(x, -1)$ (such a case could happen to to reflect uncertainty in the label of x). More generally, we will start with these strong assumptions in the bulleted list above and relax them later.

It's worth emphasizing **two main assumptions about our data** within this framework:

- We observe i.i.d training samples from (unknown) distribution D.
- Future (*unseen*) examples are drawn from the same distribution D.

The second point is easier to forget.

Expected vs. Empirical Error. Let's look a bit more closely at our objective: minimizing our *expected error*

$$L_D(h) := \mathbb{P}_{(x,y)\sim D}\{h(x) \neq y\} \tag{1.1}$$

Why not minimize this directly? Answer: we don't assume access to the data distribution D. Hence, given some sample S we use the *empirical error*:

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\{h(x_i) \neq y_i\} \tag{1.2}$$

as our proxy to D. While this is a typical setup in machine learning, it leads to the following questions:

- How should we use the empirical error?
- Is it a good estimate for the expected error? And how good?

Specifically, we are interested in their difference:

$$|L_D(h) - L_S(h)| = |\mathbb{P}_{(x,y)\sim D}\{h(x) \neq y\} - \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\{h(x_i) \neq y_i\}| \tag{1.3}$$

Please recognize that the empirical error $L_S(h)$ is a random variable as it is a function of the randomly drawn dataset $S \sim D^m$ whereas $L_D(h)$ is just a population statistic. The relationship between the two should be more clear from the below quick exercise:

$$\forall\ h : X \to \{\pm 1\} \text{ with } D \text{ over } X \times \{\pm 1\},\ \text{show that } \mathbb{E}_{S\sim D^m}[L_S(h)] = L_D(h) \tag{1.4}$$

But this is not a useful fact as its an asymptotic, and we are more interested in the difference in the case of a finite dataset size m. Thus, we often use tools like *concentration inequalities* (e.g. Hoeffding's) to create bounds like the below for some fixed h and m:

$$\mathbb{P}_{S\sim D^m}[|L_S(h) - L_D(h)| > \epsilon] \leq 2\exp(-2\epsilon^2 m) \tag{1.5}$$

Or restated equivalently (i.e. define $\delta := 2\exp(-2\epsilon^2 m)$ and "invert" the probabilities),

$$\mathbb{P}_{S\sim D^m}[|L_S(h) - L_D(h)| \leq \sqrt{\frac{\ln(2/\delta)}{2m}}] \geq 1 - \delta \tag{1.6}$$

From this expression it should be clear that as $m \to \infty$, our expected difference between expected and empirical error goes to zero.

1.3 Consistent Learning Rule Bound for Finite Hypothesis Class (Lecture 1)

Before actually creating another bound ourselves, we structure our problem even more with some prior knowledge/decisions we make:

- We restrict ourselves to a subset of functions from X to Y called our *hypothesis class* $H \subseteq Y^X$. Examples of H are given below:
  - Linear Predictors
  - Support Vector Machines (SVMs)
  - Neural Networks

H will represent our "prior knowledge" or "expert knowledge". For example, if our domain X is a set of images we would likely consider using a convolutional neural network (CNN) as our H as CNNs perform well on this kind of data.

- Assume some true function $y = f^\star(x)$ where $f^* \in H$. Our learner A will know H but not $f^\star$ (it will have to learn this function!).
- As an implication of the above assumption, we will say a sequence $((x_i, y_i))_{i=1}^m$ is *realizable* by H if the true function $f^\star \in H$ gives matching ground truth predictions[1]

Having established these assumptions, we are now ready to put them to use by creating our own bound!

Warm-up: Finite Classes. Consider the following assumptions:

- H is finite
- D is realizable by H (i.e. $\exists\ f^\star \in H$ s.t. $L_D(f^\star) = \mathbb{P}_{(x,y)\sim D}[f^\star(x) \neq y] = 0$)

Note that, as stated before, we cannot minimize $\min_{h\in H} L_D(h)$ directly and instead must work on our sample $S \sim D^m$. We present the following definition:

Definition 1.1. *We have a* consistent learning rule (CLR) *when for any input $S = \{(x_i, y_i)\}_{i=1}^m$, we can output* any $h \in H$ *s.t.* $\forall_{1\leq i\leq m}, h(x_i) = y_i$.

Then, we have the following question. If $\hat{h} := \text{CLR}_H(S)$ for some consistent learning rule CLR_H on hypothesis class H, what can we say about $L_S(\hat{h})$? It should be zero, but does this imply that $L_D(\hat{h}) = 0$?

Here's a closely-related example: consider some h where $L_D(h) = \frac{1}{2}$. This is a bad function that is correctly 50% of the time in truth. But $\mathbb{P}_{S\sim D^m}[L_S(h) = 0] > 0 \neq 0$, meaning $\exists\ S$ s.t. $L_S(h) = 0$ (i.e. we can be fooled to think h is good on some sample S.) Thus, we now **create a bound for a finite hypothesis class to control $L_D(h)$ on some CLR-learned h.**

Derivation of CLR bound for finite hypothesis class. Fix any function $h \in H$. We define ϵ s.t. $L_D(h) > \epsilon$. We proceed with the following steps:

- We first can find the probability of the bad event ($L_S(h) = 0$) below: [2]:
  $\mathbb{P}_{S\sim D^m}[L_S(h) = 0] = \prod_{i=1}^m \mathbb{P}_{S\sim D^m}\{h(x_i) = y_i\} = \prod_{i=1}^m (1 - L_D(h)) \leq (1-\epsilon)^m \leq \exp(-\epsilon m)$
- But this is just one bad function $\in H$! We can a *group* of bad functions with $B_\epsilon := \{h \in H : L_D(h) > \epsilon\} \subset H$. Then to get the probability that any CLR-learned function is "bad" we can use a *union bound*:

$$\mathbb{P}_{S\sim D^m}\big[\text{CLR}_H(S) \in B_\epsilon\big] \leq \mathbb{P}_{S\sim D^m}\big[\exists\, h \in B_\epsilon : L_S(h) = 0\big] \tag{1.7}$$

$$\leq \sum_{h\in B_\epsilon} \mathbb{P}_{S\sim D^m}\big[L_S(h) = 0\big] \leq |B_\epsilon|\, e^{-m\epsilon} \leq |H|\, e^{-m\epsilon}. \tag{1.8}$$

- We can then set $\delta := |H| \exp(-m\epsilon)$ and invert the expression to arrive at Theorem 1.2. So to ensure $\text{CLR}_H(S) \notin B_\epsilon \iff \text{CLR}_H(S) \leq \epsilon$ with probability $\geq 1 - \delta$ for some predecided $\delta \in (0, 1)$, we will need $m(\epsilon, \delta) = \frac{\ln|H| + \ln(1/\delta)}{\epsilon}$ many samples[3].

[1] Mathematically speaking, this means $\forall x_i, f^\star(x_i) = y_i \implies L_S(f^\star) = 0$. This realizability assumption is non-trivial and we will discuss it further in the course.

[2] The last argument here is done using Bernoulli's Inequality.

[3] To get this expression, solve for m in terms of δ.

Pat yourself on the back! We resummarize this bound in the following theorem:

Theorem 1.2 (CLR Bound with (ϵ, δ) fixed)**.** *For any finite class H, any (realizable) distribution D, any $(\epsilon, \delta) \in (0,1)^2$, with $m = \frac{\ln|H| + \ln(1/\delta)}{\epsilon}$, we have:*

$$\mathbb{P}_{S\sim D^m}[L_D(CLR_H(S)) \leq \epsilon] \geq 1 - \delta \tag{1.9}$$

Choosing to take the perspective that our number of samples m is fixed and so we are interested in the lowest possible error we can achieve w.h.p, we can use the following theorem:

Theorem 1.3 (CLR Bound with m fixed)**.** *For any finite class H, any (realizable) distribution D, any $\delta \in (0,1), m \in \mathbb{N}$:*

$$\mathbb{P}_{S\sim D^m}[L_D(CLR_H(S)) \leq \frac{\ln|H| + \ln(1/\delta)}{m}] \geq 1 - \delta \tag{1.10}$$

This constitutes the first learning guarantee that we have derived. Note that in our derivation we did not pay much attention to the implementation or procedure of the CLR, which will depend on H. We also used the realizability assumption, which has some implications:

- What if there is *no* predictor $h \in H$ s.t. $L_D(h) = 0$?
- In such a case, can we use with $\min_{h\in H} L_D(h)$?

In response to the second point, we will soon look at **empirical risk minimization** where:

$$\text{ERM}_H(S) = \arg\min_{h\in H} \frac{1}{|S|} \sum_{(x,y)\in S} \mathbf{1}\{h(x_i) \neq y_i\} \tag{1.11}$$

So given some function $\hat{h} := \text{ERM}_H(S)$, you might be wondering if it will satisfy our Hoeffding bound:

$$\mathbb{P}_{S\sim D^m}[|L_S(\hat{h}) - L_D(\hat{h})| \leq \sqrt{\frac{\ln(2/\delta)}{2m}}] \geq 1 - \delta \tag{1.12}$$

The answer is no. This is because that bound operates on a fixed h seen *a priori* before our sampled data, whereas $\hat{h}$ is a function of the data (e.g. $\hat{h}$ is a random variable) and so the inequality does not apply.

Note on Overfitting. Furthermore, while the set of cases where $|L_S(h) - L_D(H)| > \sqrt{\frac{\ln(2/\delta)}{2m}}$ may only have δ probability w.r.t. $S \sim D^m$, they all add up and so given many $\{h_i\}_{i=1}^K \subset H, \mathbb{P}_{S\sim D^m}[\exists\ i \text{ s.t. } |L_D(h_i) - L_S(h_i)| \text{ is large}]$ is not small. Thus, we want a stronger guarantee that w.h.p all empirical errors $L_S(h)$ are close to their expected errors $L_D(h)$:

$$\mathbb{P}_{S\sim D^m}[\forall h \in H : |L_S(h) - L_D(h)| > \epsilon] \leq \dots \tag{1.13}$$

This is known as *uniform convergence.*

1.4 Uniform Convergence & the Probably Approximately Correct (PAC) Framework (Lecture 2)

We start by giving our first attempt at a uniform convergence bound:

Theorem 1.4 (Hoeffding-derived Uniform Convergence Bound for finite H)**.** *For a finite hypothesis class $|H| < \infty$, we have the following uniform convergence bound from the a priori Hoeffding bound:*

$$\mathbb{P}_{S\sim D^m}[\exists h \in H : |L_D(h) - L_S(h)| > \epsilon] \leq |H| \cdot \mathbb{P}_{S\sim D^m}[|L_D(h) - L_S(h)| > \epsilon] = 2|H|\exp(-2\epsilon^2 m) \tag{1.14}$$

Defining $\delta := 2|H|\exp(-2\epsilon^2 m)$ and solving for m, we arrive at the data-form of this bound:

Theorem 1.5 (Theorem 1.5 for fixed (ϵ, δ))**.** *For any finite hypothesis clas $|H| < \infty$, any distribution D, any $(\epsilon, \delta) \in (0,1)^2$ we have:*

$$\mathbb{P}_{S\sim D^m}[\forall h \in H : |L_S(h) - L_D(h)| \leq \epsilon] \geq 1 - \delta \tag{1.15}$$

where $m(\epsilon, \delta) = \frac{\ln|H| + \ln(2/\delta)}{2\epsilon^2}$.

Note here that in contrast to our CLR bound Theorem 1.2, we will require $\frac{1}{\epsilon^2}$ samples as opposed to $\frac{1}{\epsilon}$. So removing the realizability assumption means that we will need more samples. Omar notes that we will explore a relaxed realizability assumption that leads to $m = O(\frac{1}{\epsilon})$ in our homework.

From these bounds, we can create some ERM-specific bounds:

Theorem 1.6 (ERM "Post-hoc" Guarantee for finite H)**.** *For any finite class H, any distribution D, any $\delta \in (0,1), m \in \mathbb{N}$, with probability $\geq 1 - \delta$ over $S \sim D^m$, we have:*

$$L_D(ERM_H(S)) \leq L_S(ERM_H(S)) + \sqrt{\frac{\ln|H| + \ln(2/\delta)}{2m}} \tag{1.16}$$

Proof. We can invoke Theorem 1.5 with $\epsilon = \sqrt{\frac{\ln|H|+\ln(2/\delta)}{2m}}$ to have $\geq 1 - \delta$ probability that $\forall h \in H : |L_D(h) - L_S(h)| \leq \epsilon$. But $\text{ERM}_H(S) \in H \implies L_D(\text{ERM}_H(S)) \leq L_S(\text{ERM}_H(S)) + \epsilon$ with $\geq 1 - \delta$ probability. So we are finished. □

Theorem 1.7 (ERM "A-Priori" Guarantee for finite H)**.** *For any finite class H, any distribution D, any $\delta \in (0,1), m \in \mathbb{N}$, with probability $\geq 1 - \delta$ over $S \sim D^m$,*

$$L_D(ERM_H(S)) \leq \min_{h\in H} L_D(h) + 2\sqrt{\frac{\ln|H| + \ln(2/\delta)}{2m}} \tag{1.17}$$

Proof. Note that by definition of ERM, $\forall\ \tilde{h} \in H, L_S(\text{ERM}_H(S)) \leq L_S(\tilde{h})$. Furthermore, $\forall\ \tilde{h} \in H : L_S(\tilde{h}) \leq L_D(\tilde{h}) + \epsilon$ with $\geq 1 - \delta$ probability. So with $\geq 1 - \delta$ probability we have:

$$\forall\ \tilde{h} \in H, \underbrace{L_D(\text{ERM}_H(S)) \leq L_S(\text{ERM}_H(S))}_{\text{see proof of Theorem 1.6}} + \epsilon \leq L_S(\tilde{h}) + \epsilon \leq L_D(\tilde{h}) + 2\epsilon$$

We apply $\min_{\tilde{h}}$ to both sides of this inequality to arrive at the theorem. □

Before moving forward, we point out the following concepts of approximation and estimation error shown in Theorem 1.7.

- The approximation error $\min_{h\in H} L_D(h)$ reduces with richer/larger hypothesis classes H
- However, these expanded hypothesis classes will demand more samples in order to maintain the same estimation error $2\sqrt{\frac{\ln|H|+\ln(2/\delta)}{2m}}$

We now move onto formally defining **Probability Approximately Correct (PAC)** learning, which Omar notes won a Turing Award. We provide the following two definitions:

Definition 1.8 (Realizably-PAC-Learnable Hypothesis Class)**.** *A hypothesis class H is realizably-PAC-learnable if there exists a learning rule A s.t. $\forall(\epsilon, \delta) \in (0,1)^2, \exists\ m(\epsilon, \delta) \in \mathbb{N}, \forall$ distributions D s.t. $\inf_{h\in H} L_D(h) = 0$,*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1 - \delta \tag{1.18}$$

Note that this type of PAC-learnable hypothesis class is *distribution independent*, meaning the bound applies for any data distribution D that is realizable (i.e. $\inf_{h\in H} L_D(h) = 0$). Dropping the realizability assumption for H, we provide another PAC definition for a hypothesis class:

Definition 1.9 (Agnostically-PAC-learnable Hypothesis Class)**.** *A hypothesis class H is agnostically-PAC-learnable if there exists a learning rule A such that $\forall(\epsilon,\delta)\in(0,1)^2, \exists\, m(\epsilon,\delta)\in\mathbb{N}, \forall$ distributions D,*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}\{L_D(A(S)) \le \inf_{h\in H} L_D(h) + \epsilon\} \ge 1-\delta \tag{1.19}$$

Looking at Definition 1.8 and Theorem 1.2, we get the following corollary.

Corollary 1.10 (Finite classes H are realizably-PAC-learnable with CLR.)**.** *All finite classes H are realizably-PAC-learnable using CLR with sample complexity*

$$m(\epsilon,\delta) = \frac{\ln|H| + \ln(1/\delta)}{\epsilon}$$

Similarly looking at Definition 1.9 and Theorem 1.7, we arrive at the following corollary:

Corollary 1.11 (Finite classes H are agnostically-PAC-learnable with ERM)**.** *All finite classes H are agnostically-PAC-learnable using ERM with sample complexity*

$$m(\epsilon,\delta) = O\left(\frac{\ln|H| + \ln(1/\delta)}{\epsilon}\right)$$

So we have established that all finite hypothesis classes are realizably and agnostically PAC-learnable. Now what about infinite classes? And can we learn with less samples than log-cardinality (i.e. $m(\epsilon,\delta) \ll \ln|H|$)? The answer is yes, and we now begin our study of the legendary VC Dimension.

1.5 Vapnik-Chervonenkis (VC) Dimension (Lecture 2)

We first start by developing some technology of the **growth function**. For $C = (x_1,\dots,x_m)\in X^m$, define the restriction (or projection) of H onto C as:

$$H|_C = \{(h(x_1),\dots,h(x_m)) \mid h\in H\} \tag{1.20}$$

We can then define the growth function as $\Gamma_H(m) = \max_{C\in X^m}|H|_C|$. We look at a few examples to understand how this growth function works:

- $X = \{1,\dots,100\}, H = \{\pm1\}^X$. Then we have $\Gamma_H(m) = \min(2^m, 2^{100})$.
- $X = \{1,\dots,2^{100}\}, H = \{\mathbf{1}[x\le\theta] \mid \theta\in\{1,\dots,2^{100}\}\}$ will have $\Gamma_H(m) = \min(m+1, 2^{100})$

The idea for both these two examples is that when two of the data points are the same (i.e. $m > |X|$), they must be labeled identically and so the growth function hits a limit. Moreover, we should observe that both function classes in these examples have the same cardinality but that the growth function $\Gamma_H(m)$ can distinguish between them (H is a lot more complex in the first example, and hence has the higher growth function.) As a teaser for future results, the growth function gives us the following result which we will later prove:

Theorem 1.12. *(Growth Function Data Bound for Realizable Distribution)*
For any hypothesis class H, any (realizable) distribution D, any $(\epsilon,\delta)\in(0,1)^2$ with sample complexity:

$$m(\epsilon,\delta) = O\left(\frac{\ln[\Gamma_h(2m)] + \ln(1/\delta)}{\epsilon}\right) \tag{1.21}$$

with probability $\ge 1-\delta$ over $S\sim D^{m(\epsilon,\delta)}$ we have that $\forall h\in H: L_S(h) = 0 \implies L_D(h)\le\epsilon$.

We also have a similar theorem in the case that D is not realizable, where $m(\epsilon, \delta) \propto \frac{1}{\epsilon^2}$:

Theorem 1.13. *(Growth Function Data Bound for any Distribution)*
For any hypothesis class H, any distribution D, any $(\epsilon, \delta) \in (0,1)^2$ with sample complexity:

$$m(\epsilon, \delta) = O\left(\frac{\ln[\Gamma_h(2m)] + \ln(1/\delta)}{\epsilon^2}\right) \tag{1.22}$$

with probability $\geq 1 - \delta$ over $S \sim D^{m(\epsilon,\delta)}$ we have that $\forall h \in H : |L_D(h) - L_S(h)| \leq \epsilon$.

The main thing to notice here is that we are no longer having our sample complexity $m(\epsilon, \delta)$ tied to $\ln|H|$. So now we do not require our bounds to be finite, and can work with infinite function classes!

Vapnik-Chervonenkis (VC) Dimension. Using the technology we have so far, we are ready to define the VC dimension. We say $C = \{x_1, \ldots, x_m\}$ is *shattered* by H if $|H|_C| = 2^m$, i.e. the projection contains all 2^m possible labelings. The VC-dimension of H, denoted by $\text{vc}(H)$ is the largest number of points that can be shattered by H:

$$\text{vc}(H) = \max\{m \in \mathbb{N} : \Gamma_H(m) = 2^m\} \tag{1.23}$$

We say $\text{vc}(H)$ is infinite if H is infinite[4] and $\forall m, \Gamma_H(m) = 2^m$. We now practice in class with a few examples of the VC Dimension:

- $X = \{1, \ldots, 100\}, \quad H = \{\pm 1\}^X$.
- $X = \{1, \ldots, 2^{100}\}, \quad H = \{\mathbf{1}[x \leq \theta] \mid \theta \in \{1, \ldots, 2^{100}\}\}$.
- $X = \mathbb{R}, \quad H = \{\mathbf{1}[x \leq \theta] \mid \theta \in \mathbb{R}\}$.
- $X = \mathbb{R}, \quad H = \{\mathbf{1}[a \leq x \leq b] \mid a, b \in \mathbb{R}\}$.
- Axis-aligned rectangles[5] (in $\mathbb{R}^d$).

Note that in order to show that the $\text{vc}(H) = k$, we must show that we can shatter some set of k points but no set of $k+1$ points.

We now transition from introducing the VC dimension to beginning a long journey to proving hypothesis classes with finite VC dimension are PAC-learnable.

1.6 PAC-Learnability of Hypothesis Classes with Finite VC Dimension (Lecture 3)

We begin with an extremely famous lemma used to bound the growth function:

Lemma 1.14. *(Sauer-Shelah-Perles Lemma) If $vc(H) = d$, then for all m:*

$$\Gamma_H(m) \leq \sum_{i=0}^{d} \binom{m}{i} \tag{1.24}$$

In particular, when $m > d$, we have $\Gamma_H(m) \leq (em/d)^d = O(m^d)$.

[4] If H is finite, $\text{vc}(H) \leq \log_2|H|$ as $\Gamma_H(m) \leq |H|$ (see definition.)

[5] **SPOILER**: Answer is four. Any set of five points will have an "interior point" in the convex hull and so you cannot make all boundary points be class one but the interior point be class zero.

Note that when $m \leq d, \Gamma_H(m) = 2^m$, which is expected by definition of the VC dimension d. The second part of the lemma, the case in which $m > d$, helps us gain more information on the growth function beyond just the fact that it is $< 2^m$. More specifically, we see that the number of "behaviors" (i.e. $|H|_C|$) is on the order of m^d for $m > d$, meaning it shifts from exponential growth (i.e. 2^m for $m \leq d$) to polynomial (i.e. m^d for $m \geq d$)[6]. We will prove this lemma by actually proving the following stronger statement:

Lemma 1.15. *(Pajor's 1985 Refinement of Sauer-Shelah-Perles Lemma) If $vc(H) = d$, then for all $C \in X^m$ we have:*

$$|H|_C| \leq |\{B \subseteq C : H \text{ shatters } B\}| \tag{1.25}$$

Note that given this refinement is true, then the fact[7] $|\{B \subseteq C : H \text{ shatters } B\}| \leq \sum_{i=0}^{d} \binom{m}{i}$ implies Lemma 1.14. So we now prove Lemma 1.15.

Proof. We prove this with induction. We start with our base case, when $m = 1$ and $C = \{x_1\}$. Then $|H|_C| = 1$ if all $h \in H$ classify x_1 the same and so only $\emptyset \in \{B \subseteq C : H \text{ shatters } B\}$ as x_1 is not shattered. If $|H|_C| = 2$, then $\{x_1\}$ is shattered and so $\{B \subseteq C : H \text{ shatters } B\} = \{\emptyset, \{x_1\}\}$. So in both cases, we have a strict equality. Thus the base case is satisfied. We now proceed with the inductive step, assuming that this inequality holds for all $k < m$, and we WTS it holds for m. We start by defining $C = \{x_1, \ldots, x_m\} \in X^m$ and $C' = \{x_2, \ldots, x_m\}$, which is just C with x_1 removed. Now consider:

$$A = \{(y_2, \ldots, y_m) : (+1, y_2, \ldots, y_m) \in H|_C \text{ or } (-1, y_2, \ldots, y_m) \in H|_C\}$$

and

$$B = \{(y_2, \ldots, y_m) : (+1, y_2, \ldots, y_m) \in H|_C \text{ and } (-1, y_2, \ldots, y_m) \in H|_C\}$$

Then first note that each element in A/B contributes only one labeling to $H|_C$ whereas each element in B contributes two labelings to $H|_C$. Thus, $|H|_C| = |A/B| + 2|B| = |A| + |B|$. Also observe that $A = H|_{C'}$ as $A \subseteq H|_{C'}$ by construction and every $(y_2, \ldots, y_m)$ labeling in $H|_{C'}$ is in A. By our induction hypothesis, it holds that:

$$|A| = |H|_{C'}| \leq |\{S \subseteq C' : H \text{ shatters } S\}| = |\{S \subseteq C : x_1 \notin S \text{ and } H \text{ shatters } S\}|$$

We can then define $H' \subseteq H$ as the set of all functions with a "twin" for labeling x_1:

$$H' = \{h \in H : \exists\ h' \in H \text{ s.t. } h'(x_1) = 1 - h(x_1) \text{ and } \forall i\ \in [2, m], h'(x_i) = h(x_i)\}$$

Quite elegantly, we have $B = H'|_{C'}$ as it is the labelings of C' generated from only those functions in H' which would produce both labels on $x_1 \in C$ (i.e. the set of functions H'). Furthermore, note that if H' shatters any subset of C', that means it could shatter that subset *and* x_1. So we apply our induction hypothesis once more:

$$\begin{aligned} |B| = |H'|_{C'}| \leq |\{S \subseteq C' : H' \text{ shatters } S\}| = |\{S \subseteq C' : H' \text{ shatters } S \cup \{x_1\}\}| \\ = |\{S \subseteq C : x_1 \in S \text{ and } H' \text{ shatters } S\}| \leq |\{S \subseteq C : x_1 \in S \text{ and } H \text{ shatters } S\}| \end{aligned}$$

We can combine the above two inequalities for $|A|$ and $|B|$ to arrive at:

$$\begin{aligned} |H|_C| = |A| + |B| \leq |\{S \subseteq C : x_1 \notin S \text{ and } H \text{ shatters } S\}| + |\{S \subseteq C : x_1 \in S \text{ and } H \text{ shatters } S\}| \\ = |\{S \subseteq C : H \text{ shatters } S\}| \end{aligned}$$

which finishes the proof. □

[6] Omar notes that this is what makes learning possible. This statement should make more sense by the end of this lecture.

[7] This inequality is true because $\sum_{i=0}^{d} \binom{m}{i}$ gives the number of subsets of valid shatterable size (i.e. $1, \ldots, d$) of m points, which in our case is C.

So applying the Sauer-Shelah-Perles lemma (for when $m > d$) to (1) the growth function sample complexity bound for realizable distributions stated in Theorem 1.12 and (2) the distribution-agnostic growth function sample complexity bound stated in Theorem 1.13 , we arrive at the following corollary:

Corollary 1.16. *(Infinite Hypothesis Classes with finite VC Dimension are PAC-learnable.) Any hypothesis class H with finite VC dimension is:*

1. *Realizably-PAC-learnable using ERM with sample complexity:*

$$m(\epsilon, \delta) = O\left(\frac{vc(H)\ln(1/\epsilon) + \ln(1/\delta)}{\epsilon}\right) \tag{1.26}$$

2. *Agnostically-PAC-learnable using ERM with sample complexity:*

$$m(\epsilon, \delta) = O\left(\frac{vc(H) + \ln(1/\delta)}{\epsilon^2}\right) \tag{1.27}$$

This is a big result in statistical learning theory. The realizable sample complexity bound above was derived from Theorem 1.12, so now is probably a good time to actually prove Theorem 1.12. We will not prove the agnostic case bound in Theorem 1.13 as it is similar. We restate the theorem below (boxed), in the version we want to prove, before starting its (long) proof[8]:

$$\boxed{\mathbb{P}_{S\sim D^m}[\forall h \in H : L_S(h) = 0 \implies L_D(h) \leq 2\frac{\ln[\Gamma_H(2m)] + \ln(2/\delta)}{m}] \geq 1 - \delta} \tag{1.28}$$

Proof. We first define $\epsilon = 2\frac{\ln[\Gamma_H(2m)]+\ln(2/\delta)}{m}$. Given a set $S = \{(x_i, y_i)\}_{i=1}^m$ of m examples, define the event

$$A_S = \{\exists h \in H : L_D(h) > \epsilon \wedge L_S(h) = 0\}$$

Our goal is to show that $\mathbb{P}_{S\sim D^m}[A_S] \leq \epsilon$. Now let us consider drawing two sets S, S' of m examples each. We can define the event:

$$B_{S,S'} = \{\exists\ h \in H : L_{S'}(h) > \frac{\epsilon}{2} \wedge L_S(h) = 0\}$$

Now note that $\mathbb{P}_{S,S'\sim D^m}[B_{S,S'}]$ is $\geq \frac{1}{2}\mathbb{P}_{S\sim D^m}[A_S]$. The reason for this is that $\mathbb{P}_{S,S'\sim D^m}[B_{S,S'}] = \mathbb{P}_{S\sim D^m}[A_S] \cdot \mathbb{P}_{S,S'\sim D^m}[B_{S,S'} \mid A_S]$ and $\mathbb{P}_{S,S'\sim D^m}[B_{S,S'} \mid A_S] \geq \frac{1}{2}$ by a Chernoff bound as long as $m > 8/\epsilon$. Thus, given this fact note $\mathbb{P}_{S',S\sim D^m}[B_{S,S'}] \leq \frac{\delta}{2}$ implies our goal.

Now consider another experiment where we draw a fixed set S'' of $2m$ examples, and then (randomly) partition S'' into two sets S, S' each of cardinality m. Now define another event:

$$C_{S'',S',S} = \{\exists h \in H : L_{S'}(h) > \frac{\epsilon}{2} \wedge L_S(h) = 0\}$$

It should be intuitive to see that $\mathbb{P}_{S''\sim D^{2m}}[C_{S'',S',S'}] = \mathbb{P}_{S,S'\sim D^m}[B_{S,S'}]$ and so it suffices to show $\mathbb{P}_{S''\sim D^{2m}}[C_{S'',S',S'}] \leq \delta/2$. Note that for a fixed S'', this is equivalent to showing that $\mathbb{P}_{S,S'\sim D^m}[C_{S'',S',S}] \leq \delta/2$. To do so consider the following:

1. With S'' fixed, we only need to consider the projection of H onto the x's (i.e. datapoints) that appear in S''. So there are at most $\Gamma_H(2m)$ labelings we need to consider, as $\Gamma_H(2m)$ is the maximum number of labelings for any set of $2m$ datapoints, such as S''.

2. For each such labeling, we need to show that the probability of being perfect on S but have error $\geq \epsilon/2$ on S' is low. We will then union bound.

[8] Omar presented a brief primer on the Chernoff concentration inequality before starting this proof, which might be helpful.

We proceed by fixing a labeling $h \in H|_{S''}$. We can assume that h makes at least $\frac{\epsilon m}{2}$ mistakes on S'', as otherwise $\mathbb{P}_{S''\sim D^{2m}}[C_{S'',S',S'}] = 0 \leq \delta/2$ and so our proof is finished. Now when we randomly split S'' into S and S', what is the chance that all these mistakes land in S? Consider the following analogy. We can partition S'' by randomly pairing the points together $(a_1, b_1), \ldots, (a_m, b_m)$. Then, for each pair (a_i, b_i) we can flip a coin where (i) heads mean a_i goes to S and b_i goes to S' and (ii) tails mean vice versa. Observe that if there is any pair (a_i, b_i) in which h makes a mistake on both of them then the chance that all htese mistakes land in S is zero. Otherwise, the probability that all mistakes land in S is at most $(\frac{1}{2})^{\epsilon m/2}$, as there is a 50% chance of each of the $\geq \frac{\epsilon m}{2}$ mistakes landing in S based on this coin flip procedure.

Because this was for an arbitrary labeling, we apply a union bound over all possible labelings in $H|_{S''}$:

$$\mathbb{P}_{S''\sim D^{2m}}[C_{S'',S',S}] \leq \Gamma_H(2m) \cdot 2^{-\epsilon m/2}$$

and so to conclude the proof just solve for $m(\epsilon, \delta)$ s.t. $\Gamma_H(2m) \cdot 2^{-\epsilon m/2} \leq \delta/2$. □

Continuing forward with our exploration, we are interested in the following questions:

1. Are there classes with *infinite* VC dimension that are PAC-learnable?
2. Can we learn with *fewer* samples than VC dimension?
3. Are there any hypothesis classes that are not PAC-learnable?

We now tackle the first question through the *statistical no-free-lunch (NFL) theorem*, which shows that no hypothesis class with infinite VC dimension is PAC-learnable.

Theorem 1.17 (Statistical No Free Lunch)**.** *For any hypothesis class H, any learning rule A, and any $\epsilon < 1/4$, there exists a (realizable) distribution D such that if*

$$m < \frac{vc(H) - 1}{8\epsilon} \tag{1.29}$$

then

$$\mathbb{E}_{S\sim D^m}[L_D(A(S))] \geq \epsilon \tag{1.30}$$

Note that this gives us a VC-dimension based lower bound on the required sample complexity $m(\epsilon, \delta)$ for $\mathbb{E}_{S\sim D^m}[L_D(A(S))] < \epsilon$. Thus, this means $m(\epsilon, \delta)$ is bounded both above (i.e. Corollary 1.16) and below by something $\propto \text{vc}(H)$, or, equivalently, that our sample complexity bounds in Corollary 1.16 should really be of the $\Theta(\ldots)$ kind as opposed to $O(\ldots)$.

Proof. Pick $d = \text{vc}(H)$ shatterable points $x_1, \ldots, x_d$. Then define discrete distribution P with probability mass $1 - 4\epsilon$ on x_1 and mass $\frac{4\epsilon}{d-1}$ on all other points. Now because these d points are shatterable $\implies$ we have 2^d behaviors from H $\implies$ we have 2^d possible target functions. Now pick a random labeling from the 2^d possible target functions. Then from definition of L_D we have:

$$\mathbb{E}_{S\sim D^m}[L_D(A(S))] = \mathbb{P}[\text{mistake on some test point}] \geq \frac{1}{2}\mathbb{P}[\text{test point} \notin S]$$

where the last inequality is because $\mathbb{P}[\text{mistake on test point}] = \mathbb{P}[\text{test point} \notin S \wedge \text{mistake on test point}]$ and $\mathbb{P}[\text{mistake on test point}] \geq 0.5$ as we have given this test point a randomly chosen label (you can't predict better than 50%). Continuing forward we have:

$$\mathbb{P}[\text{test point} \notin S] \geq \sum_{i=2}^{d} \mathbb{P}[\text{test point is } x_i \wedge x_i \notin S] = \sum_{i=2}^{d} \mathbb{P}[\text{test point is } x_i] \cdot \mathbb{P}[\text{test point} \notin S \mid \text{test point is } x_i]$$

$$= \sum_{i=2}^{d} \frac{4\epsilon}{d-1} \cdot (1 - \frac{4\epsilon}{d-1})^m = 4\epsilon(1 - \frac{4\epsilon}{d-1})^m \geq 4\epsilon(1 - \frac{4m\epsilon}{d-1}) \geq 4\epsilon(1 - \frac{1}{2}) = 2\epsilon$$

The first inequality comes from the fact we are not considering the case in which the test point is x_1. Furthermore, note the use of Bernoulli's inequality in the penultimate inequality and our assumed bound on m in the last inequality. From this, we have $\mathbb{E}_{S\sim D^m}[L_D(A(S))] \geq \epsilon$. □

So we can present the following theorem, which is a restatement of all we have shown so far:

Theorem 1.18 (Fundamental Theorem of Statistical Learning)**.** *For any hypothesis class H with finite VC dimension $d = vc(H)$ and any $\varepsilon, \delta \in (0,1)$:*

- *H is (realizably)-PAC-learnable with sample complexity:*

$$m(\varepsilon, \delta) = \Theta\left(\frac{d + \ln(1/\delta)}{\varepsilon}\right).$$

- *H is (agnostically)-PAC-learnable with sample complexity:*

$$m(\varepsilon, \delta) = \Theta\left(\frac{d + \ln(1/\delta)}{\varepsilon^2}\right),$$

- *H satisfies uniform convergence with sample complexity:*

$$m(\varepsilon, \delta) = \Theta\left(\frac{d + \ln(1/\delta)}{\varepsilon^2}\right).$$

We will next time start with understanding what it means for something to be *efficiently* PAC-learnable.

1.7 Efficient PAC Learnability (Lecture 4)

We give the following first *attempt* at defining what it could mean for a hypothesis class to be efficiently PAC learnable:

Definition 1.19 (Efficiently PAC-Learnable Definition Attempt)**.** *A hypothesis class H is efficiently-realizably-PAC-learnable if there exists a poly-time computable learning rule A such that $\forall(\epsilon, \delta) \in (0,1)^2, \exists\, m(\epsilon, \delta) \in \mathbb{N}$ where $\forall D$ s.t. $\inf_{h\in H} L_D(h) = 0$ we have:*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1 - \delta \tag{1.31}$$

The natural question here is *what* the runtime is polynomial in? Is $A(S)$ polynomial in the dataset size $|S|$? But this is a "cheatable" definition, as if the runtime of A is $\propto 2^{|S|}$ we can simply just draw $m'(\epsilon, \delta) := 2^{m(\epsilon,\delta)}$ many samples but have A only use $\log_2 m'(\epsilon, \delta)$ of them. So the runtime of A appears to be $\propto m'(\epsilon, \delta) = |S|$ now. Or do we mean the runtime should be polynomial in ϵ^{-1} or δ^{-1}? What we decide that we really want is that the runtime of A should be polynomial in "the size of the problem", which we will be clear shortly.

To understand this, we will start by studying a family of hypotheses classes $\{H_n\}_{n\in\mathbb{N}}$ over $\{X_n\}_{n\in\mathbb{N}}$. Usually X_n grows with n, such as $X_n = \{0,1\}^n$ or $X_n = \mathbb{R}^n$, but sometimes X_n is fixed (e.g. $X_n = \mathbb{R}^d$). Note that regardless, we are using binary labels $Y = \{\pm 1\}$. With this construction, we can give a more precise definition of efficiently PAC-learnable:

Definition 1.20 (Efficiently PAC-Learnable over hypothesis class family)**.** *A family $\{H_n\}_n$ is efficiently-realizably-PAC-learnable if there exists a learning rule A such that $\forall n, \forall(\epsilon, \delta) \in (0,1)^2, \exists\, m(\epsilon, \delta) \in \mathbb{N}$ where $\forall D$ s.t. $\inf_{h\in H} L_D(h) = 0$ we have:*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1 - \delta \tag{1.32}$$

and A can be computed in runtime $poly(n, 1/\epsilon, \log[1/\delta])$.

The first thing we should notice is that this definition clearly necessitates A to scale polynomially with the problem complexity (i.e. $n, 1/\epsilon, \log[1/\delta]$ are all proportional to the difficulty of the problem.) Another thing is that because A's runtime is bounded below by the number of samples, if $\{H_n\}_n$ is efficiently PAC-learnable $\implies m(\epsilon, \delta) \leq \text{poly}(n, 1/\epsilon, \log[1/\delta])$.

We now consider another perspective. Rather than thinking about our algorithm A as taking samples S, imagine our algorithm has access to an $O(1)$ sampling oracle of our data distribution D. In math, we think of this as $A(D, \epsilon, \delta)$ where A can sample from D in unit time using this "oracle." Taking this perspective, $m(\epsilon, \delta)$ is then the number of times A has to use this oracle to arrive at $\leq \epsilon$ expected risk with $\geq 1 - \delta$ confidence. With this perspective, we see that there is still room in Definition 1.22 for multiple views on what A's output should be:

- View 1: $A(\cdot)$ outputs a program[9] that maps $X_n \to Y$ that will run in $\text{poly}(n, 1/\epsilon, \log[1/\delta])$.
- View 2: $A(S, x)$ or $A(D, \epsilon, \delta, x)$ will output prediction $y = h(x)$.
- View 3: $A(\cdot)$ outputs a description of $h \in H$ (e.g. a decision tree.) Formally speaking, we assume A outputs $w \in \{0,1\}^\star$ with description length $|w| \leq \text{poly}(n, 1/\epsilon, \log[1/\delta])$ where there exists a poly-time algorithm $B(w, x) \mapsto h_w(x)$.

So the kinds of things we are looking at when talking about an efficiently PAC-learnable algorithm A are its runtime to produce a function/program, the runtime of this program, and/or the description length of this program.

You might be wondering what hypotheses classes will satisfy Definition 1.22, meaning that they are efficiently PAC learnable. For now, we will worry about the case in which the hypothesis class is realizable. To start, Omar notes that halfspaces are efficiently PAC-learnable, which can be shown through convex/linear optimization. If we consider polynomials as nonlinear feature maps, then through a "degree-lift" of the halfspaces, we can show that polynomials are efficiently PAC learnable. We finish this discussion by focusing on the family of conjunction hypothesis classes, which we introduce now[10]:

$$X_n = \{0,1\}^n, \quad H_n = \text{CONJ}_n = \{x \mapsto (\wedge_{j \in J} x(j)) \wedge (\wedge_{i \in I} \bar{x}(i)) \mid J, I \subseteq [n]\}, \quad Y_n = \{\pm 1\}$$

So for a fixed n, our instance space is a binary string of length n and our hypothesis class is an AND (i.e. $\wedge$) of ANDs over indices I and ANDs over indices J. As an example, we could have $h(x) = x(5) \wedge x(12) \wedge \bar{x}(7) \in H_n$. A natural question then is what the VC dimension of H_n is. For this, note that the size of H_n is 3^n and so[11]:

$$\text{vc}(H_n) \leq \log |H_n| = \log(3^n) = O(n)$$

As per our VC realizable data bound Theorem 1.12, this means that we can learn with $m(\epsilon, \delta) = O(\frac{n + \log(1/\delta)}{\epsilon})$ samples. We now show that we can efficiently learn on this task through the following $O(mn)$ algorithm:

- We take input $S = \{(x_i, y_i)\}_{i=1}^m$ of m samples
- We first initialize[12] $h = \wedge_{i=1}^n [x(i) \wedge \bar{x}(i)] \in H_n$
- Now for the ith sample, we check if $y_i = 1$. If so then for $1 \leq j \leq n$:
  - Define x_i to be the ith sample. If $x_i(j) = 1$, we will remove $\bar{x}(j)$ from h as its presence causes $h(x) = 0 \neq y_i = 1$.

[9] The word "program" is defined here through the Turing Machine terminology.

[10] Notation: $\bar{x} := 1 - x$, and $x(i) \in \{0,1\}$ gives the ith index $x \in X_n = \{0,1\}^n$.

[11] The reason $|H_n| = 3^n$ is that for each index $i \in [1, n]$, when constructing some function $h \in H_n$ we have three options: (1) include $x(i)$ in h, (2) include $\bar{x}(i) \in h$, (3) neither do (1) nor (2). Note that doing both (1) and (2) will lead to $h = 0$, and hence it is not an option.

[12] Note that this is just taking $I = J = [n]$.

– Similarly if $x_i(j) = 0$, we will remove $x(j)$ from h.

Note that this algorithm after running *will* result in an $h \in H_n$ with $L_S(h) = 0$. This should remind us of a type of algorithm we have looked at before – a consistent learning rule (Definition 1.1)! Observe that in showing that hypothesis class CONJ_n is efficiently PAC learnable, we have used the following recipe:

Theorem 1.21 (CLR Recipe to conclude that something is efficiently PAC-learnable)**.** *If $vc(H_n) \leq poly(n)$ and there is a poly-time algorithm implementing the CLR for $\{H_n\}_n$, then we have that $\{H_n\}_n$ is efficiently-PAC-learnable.*

Proof. If $\text{vc}(H_n) \leq \text{poly}(n) \implies m(\epsilon, \delta) \leq \text{poly}(n)$. Furthermore, the runtime of the CLR in this case (which will by definition will satisfy all $\leq \epsilon, \geq 1 - \delta$ constraints) is $\leq \text{poly}(n)$. So our algorithm will take $\leq \text{poly}(n)$ to read all samples and $\leq \text{poly}(n)$ to give the CLR-learned function $\implies$ the algorithm satisfies all efficient PAC-learnable constraints in $\leq \text{poly}(n)$ time $\implies \{H_n\}_n$ is efficiently PAC-learnable. □

But perhaps the converse question is more interesting:

$$\text{If } \forall n, H_n \text{ is efficiently-PAC-learnable } \implies \text{ is there a poly-time CLR for } H_n?$$

We will return to this question later. We move onto considering a more complex 3-Term DNF function class H_n:

$$H_n = \{T_1 \vee T_2 \vee T_3 | T_1, T_2, T_3 \in \text{CONJ}_n\} \tag{1.33}$$

As an example of a function in this hypothesis class:

$$h(x) = (x_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6) \vee (x_7 \wedge x_8 \wedge x_9 \wedge \bar{x}_{10}) \in H_n$$

We once again have:

$$\text{vc}(H_n) \leq \log |H_n| \leq \log[(3^n)^3]$$

So we can learn this problem with $\text{poly}(n, 1/\epsilon, \log[1/\delta])$ samples. *But can we do so efficiently?* We start with the claim that finding a consistent hypothesis in H_n is NP-hard[13]. Thus, assuming that P $\neq$ NP, this means that there is no polynomial time algorithm for deciding[14] if there is a CLR on H_n. Keeping this thought in mind, we introduce efficient *proper* PAC-learning:

Definition 1.22 (Efficiently Properly PAC-Learnable Hypothesis Class Family $\{H_n\}_n$)**.** *A family $\{H_n\}_n$ is efficiently-properly-PAC-learnable if there exists a learning rule A such that $\forall n, \forall (\epsilon, \delta) \in (0,1)^2, \exists\, m(\epsilon, \delta) \in \mathbb{N}$ where $\forall D$ s.t. $\inf_{h \in H} L_D(h) = 0$ we have:*

$$\mathbb{P}_{S \sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1 - \delta \tag{1.34}$$

where A can be computed in time $poly(n, 1/\epsilon, \log[1/\delta])$, and A always outputs a predictor in H_n.

We will now take a look at the algorithmic hardness of proper learning on any hypothesis class family. We start with the decision task of seeing if a consistent hypothesis on a given dataset.

[13] Omar gives a very involved proof in the slides, which reduces the problem to the problem of graph 3-colorability, which is NP-complete.

[14] Recall that if finding something is NP-hard, then the corresponding decision task will necessarily also be NP-hard.

Hardness of Proper Learning. For a family $\{H_n\}_n$ consider the following decision problem:

$$\text{CONS}_{H_n}(S) = 1 \iff \exists\, h \in H_n \text{ s.t. } L_S(h) = 0$$

We present the following theorem:

Theorem 1.23. *If H_n over $X_n = \{0,1\}^n$ is efficiently-properly-PAC-learnable then $CONS_{H_n}(S) \in RP$.*

In words, we are saying that if any arbitrary H_n over our binary string instance space X_n is efficiently-properly-PAC-learnable, then there exists a (randomized) algorithm B with polynomial runtime in n where for any input dataset S we have:

$$\mathbb{P}[B(S) = 1 \mid \text{CONS}_{H_n}(S) = 1] \geq \frac{7}{8} > \frac{1}{2}, \quad \text{and } \mathbb{P}[B(S) = 0 \mid \text{CONS}_{H_n}(S) = 0] = 0$$

Proof. Let A be an efficient-proper-PAC-learning algorithm for H_n. Now consider the following algorithm B upon receiving input S:

1. Define $\hat{h} := A(D, \epsilon, \delta) \in H_n$, where $D = \text{Unif}(S), \epsilon = \frac{1}{2|S|}, \delta = \frac{1}{8}$. Such choice for D means $L_D(\hat{h}) = L_S(\hat{h})$.
2. Check if $L_S(\hat{h}) = 0$ by evaluating $\hat{h}$ on all $(x, y) \in S$.
3. Return $1_{(L_S(\hat{h})=0)}$.

Time to analyze this algorithm B! First note if $\text{CONS}_{H_n}(S) = 0 \implies B$ outputs 0 (since in this case $\forall h \in H_n, L_S(h) > 0$). Now assume $\text{CONS}_{H_n}(S) = 1$. Then with probability $\geq 1 - \delta = 7/8$, $L_D(\hat{h}) = L_S(\hat{h}) \leq \epsilon \leq \frac{1}{2|S|}$. But $L_S(\hat{h})$ cannot be nonzero and smaller than $\frac{1}{|S|} \implies L_S(\hat{h}) = 0$. In this case, B would return 1.

Note that because A is an efficient-proper-PAC-learning algorithm, the runtime of B (our randomized algorithm to solve $\text{CONS}_{H_n}(S)$) is polynomial in n and $\frac{1}{\epsilon} \sim |S|$. Thus $\text{CONS}_{H_n}(S) \in \text{RP}$. □

So switching back to our example where H_n is a 3-term DNF, we have proved the following corollary:

Corollary 1.24. *If RP ≠ NP, then H_n (3-term DNF) is not efficiently properly PAC-learnable.*

Proof. BWOC, if H_n (3-term-DNF) is efficiently properly PAC-learnable $\implies \text{CONS}_{H_n}(S) \in \text{RP} \implies$ 3COLOR $\in$ RP $\implies$ RP = NP, which is a contradiction. □

Now assuming RP ≠ NP, we restate Theorem 1.21 and Theorem 1.23 for any arbitrary family $\{H_n\}_n$:

1. If $\text{vc}(H_n) \leq \text{poly}(n)$ and there is a poly-time algorithm implementing consistent learning $\implies H_n$ is efficiently-properly-PAC-learnable.
2. If solving $\text{CONS}_{H_n}(S)$ is NP-hard $\implies$ the family $\{H_n\}_n$ is *not* efficiently-properly-PAC-learnable.

What about *improper* learning algorithms, where we allow the algorithm to output something outside the hypothesis class? Furthermore, if a family H_n is efficiently-properly-PAC-learnable, what can we say about $\tilde{H}_n \subset H_n$? Is $\tilde{H}_n$ efficiently-properly-PAC learnable? We continue this exploration further with our same 3-term DNF H_n family. Recall:

$$H_n = \{T_1 \vee T_2 \vee T_3 | T_1, T_2, T_3 \in \text{CONJ}_n\}$$

Note that for an example function in this class:

$$h(x) \;=\; (x_1 \wedge x_2 \wedge x_3) \;\vee\; (\bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6) \;\vee\; (x_7 \wedge x_8 \wedge x_9 \wedge \bar{x}_{10}).$$

can also be represented as:

$$
\begin{aligned}
h(x) \;= &(x_1 \vee \bar{x}_4 \vee x_7) \wedge (x_1 \vee \bar{x}_4 \vee x_8) \wedge (x_1 \vee \bar{x}_4 \vee x_9) \wedge (x_1 \vee \bar{x}_4 \vee \bar{x}_{10}) \\
&\wedge (x_1 \vee \bar{x}_5 \vee x_7) \wedge (x_1 \vee \bar{x}_5 \vee x_8) \wedge (x_1 \vee \bar{x}_5 \vee x_9) \wedge (x_1 \vee \bar{x}_5 \vee \bar{x}_{10}) \\
&\wedge (x_1 \vee \bar{x}_6 \vee x_7) \wedge (x_1 \vee \bar{x}_6 \vee x_8) \wedge (x_1 \vee \bar{x}_6 \vee x_9) \wedge (x_1 \vee \bar{x}_6 \vee \bar{x}_{10}) \\
&\wedge (x_2 \vee \bar{x}_4 \vee x_7) \wedge (x_2 \vee \bar{x}_4 \vee x_8) \wedge (x_2 \vee \bar{x}_4 \vee x_9) \wedge (x_2 \vee \bar{x}_4 \vee \bar{x}_{10}) \\
&\wedge (x_2 \vee \bar{x}_5 \vee x_7) \wedge (x_2 \vee \bar{x}_5 \vee x_8) \wedge (x_2 \vee \bar{x}_5 \vee x_9) \wedge (x_2 \vee \bar{x}_5 \vee \bar{x}_{10}) \\
&\wedge (x_2 \vee \bar{x}_6 \vee x_7) \wedge (x_2 \vee \bar{x}_6 \vee x_8) \wedge (x_2 \vee \bar{x}_6 \vee x_9) \wedge (x_2 \vee \bar{x}_6 \vee \bar{x}_{10}) \\
&\wedge (x_3 \vee \bar{x}_4 \vee x_7) \wedge (x_3 \vee \bar{x}_4 \vee x_8) \wedge (x_3 \vee \bar{x}_4 \vee x_9) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_{10}) \\
&\wedge (x_3 \vee \bar{x}_5 \vee x_7) \wedge (x_3 \vee \bar{x}_5 \vee x_8) \wedge (x_3 \vee \bar{x}_5 \vee x_9) \wedge (x_3 \vee \bar{x}_5 \vee \bar{x}_{10}) \\
&\wedge (x_3 \vee \bar{x}_6 \vee x_7) \wedge (x_3 \vee \bar{x}_6 \vee x_8) \wedge (x_3 \vee \bar{x}_6 \vee x_9) \wedge (x_3 \vee \bar{x}_6 \vee \bar{x}_{10})
\end{aligned}
$$

So based on this representation, we are making the following claim:

$$\text{3-term DNF}_n \subset \text{3CNF}_n = \{\wedge_{i=1}^{r} L_i \mid L_i \text{ is a disjunction of 3 variables, and } r \in \mathbb{N}\}$$

We now make a second claim, that 3CNF_n is efficiently-properly-PAC-learnable. Our reasoning for this is as follows:

- There are $(2n)^3$ possible disjunctions of 3 variables (i.e. the number of L_i's, or r)
- Treat each of these as a single new variable in a lifted space. Similar to defining a feature map: $\phi(x) = (x_1 \vee x_2 \vee x_3, \ldots, \bar{x}_{n-2} \vee \bar{x}_{n-1} \vee \bar{x}_n)$.
- We now run the algorithm for consistently solving conjunctions in the lifted space.
- We have $O(n^3)$ many variables in this lifted space and a sample complexity of $O(n^3/\epsilon)$ (recall the VC dimension on CONJ_n is n so sample complexity $O(n/\epsilon)$ required.) Thus the runtime of using this algorithm is $O(n^3) * O(n^3/\epsilon) = O(n^6/\epsilon)$, which is polynomial in n and $1/\epsilon$.

This gives us the following corollary:

Corollary 1.25. *3-term DNF_n is efficiently-improperly-PAC-learnable.*

The reason for this is because we can learn any 3-term DNF consistently, except that we will get a 3-term CNF, which is not necessarily a 3-term DNF. Furthermore, note that the cardinality bound on the VC dimension gives us a theoretical sample complexity of $O(n/\epsilon)$ for the 3-term DNF_n. However, finding a consistent hypothesis in the 3-term DNF_n family is NP-hard. But by transforming 3-term DNF_n into the 3-term CNF_n family, we can find a consistent hypothesis but with a larger sample complexity of $O(n^3/\epsilon)$. This is a *computational-statistical tradeoff*: we have to use more computation in order to achieve better statistical results.

We finally finish this lecture with a discussion of our previously posed $H_n \subseteq H'_n$ question:

- If $H_n \subseteq H'_n$, then
  - If H'_n is efficiently-PAC-learnable,
    - then H_n is efficiently-PAC-learnable.
  - If H_n is not efficiently-PAC-learnable,
    - then H'_n is not efficiently-PAC-learnable.
  - If H_n is not efficiently-properly-PAC-learnable,
    - then H'_n is ???.

As an ending question, we will want to consider if there are any hypotheses classes that are not efficiently PAC-learnable, even improperly. If they do exist, how would we prove that they are not efficiently PAC-learnable?