Yale University S&DS 669
Statistical Learning Theory

Instructor: Dr. Omar Montasser
Scribe: Anish Lakkapragada

November 5, 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.

[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.

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*:

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

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

- We can then set $\delta := |H| \exp(-m\epsilon)$ and invert the expression to arrive at Theorem 1.2. So to ensure $\mathrm{CLR}_H(S) \notin B_\epsilon \iff \mathrm{CLR}_H(S) \leq \epsilon$ 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].

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 $\min_{h\in H} L_D(h)$?

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

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

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,*

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

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 $C = (x_1, \ldots, x_m) \in X^m$, define the restriction (or projection) of H onto C as:

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

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:

- $X = \{1, \ldots, 100\}, H = \{\pm 1\}^X$. Then we have $\Gamma_H(m) = \min(2^m, 2^{100})$.
- $X = \{1, \ldots, 2^{100}\}, H = \{\mathbf{1}[x \leq \theta] \mid \theta \in \{1, \ldots, 2^{100}\}\}$ 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 $\geq 1 - \delta$ over $S \sim D^{m(\epsilon,\delta)}$ we have that $\forall h \in H : L_S(h) = 0 \implies L_D(h) \leq \epsilon$.

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:

[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.

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 $\Gamma_H(m) \leq (em/d)^d = O(m^d)$.

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:

[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.

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

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:

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

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

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

which finishes the proof. □

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]:

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

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

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.

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 $\Theta(\dots)$ kind as opposed to $O(\dots)$.

Proof. Pick $d = \text{vc}(H)$ shatterable points $x_1, \dots, x_d$. 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.

Chapter 2

Computational Aspects of PAC Learning

2.1 Efficient PAC Learnability (Lecture 4 & 5)

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

Definition 2.1 (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:*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1-\delta \tag{2.1}$$

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 2.2 (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:*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1-\delta \tag{2.2}$$

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 2.2 for multiple views on what A's output should be:

- View 1: $A(\cdot)$ outputs a program[1] 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 2.2, 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[2]:

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

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[3]:

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

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

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

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

[3] 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.

- We first initialize[4] $h = \wedge_{i=1}^{n}[x(i) \wedge \bar{x}(i)] \in H_n$
- 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 $h(x) = 0 \neq y_i = 1$.
  - 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 2.3 (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:

If $\forall n, H_n$ is efficiently-PAC-learnable $\implies$ 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:

$$H_n = \{T_1 \vee T_2 \vee T_3 | T_1, T_2, T_3 \in \text{CONJ}_n\} \tag{2.3}$$

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[5]. Thus, assuming that P $\neq$ NP, this means that there is no polynomial time algorithm for deciding[6] if there is a CLR on H_n. Keeping this thought in mind, we introduce efficient *proper* PAC-learning:

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

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

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

Definition 2.4 (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:*

$$\mathbb{P}_{S \sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \epsilon] \geq 1 - \delta \tag{2.4}$$

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.

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 2.5. *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:

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

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 $\text{CONS}_{H_n}(S) \in$ RP. □

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

Corollary 2.6. *If RP $\neq$ 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 $\implies$ $\text{CONS}_{H_n}(S) \in \text{RP} \implies \text{3COLOR} \in \text{RP} \implies \text{RP} = \text{NP}$, which is a contradiction. $\square$

Now assuming RP $\neq$ NP, we restate Theorem 2.3 and Theorem 2.5 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 2.7. *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?

Hardness of Improper Learning. In general, to conclude that a family of hypotheses classes is not efficiently-PAC-learnable (even improperly), we must show that this family can solve a cryptographic problem that is assumed to be computationally intractable. For example, assuming the cryptographic "Discrete Cube Root" is computationally intractable, then the class of log-depth polynomial-size circuits (AND/OR networks) will not be efficiently-PAC-learnable as if it was, it would imply that this cryptographic problem is tractable (which is a contradiction by our assumption.)

We now move onto efficient-proper *agnostic* learning, where we do not assume distributions to be realizable. We present the following definition, which is the agnostic counterpart to Definition 2.4:

Definition 2.8 (Efficiently Properly Agnostic PAC-Learnable family $\{H_n\}_n$)**.** *A family $\{H_n\}_n$ is efficiently-properly-PAC-learnable in agnostic setting 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 $\forall D$ we have:*

$$\mathbb{P}_{S \sim D^{m(\epsilon,\delta)}}[L_D(A(S)) \leq \inf_{h \in H_n} L_D(h) + \epsilon] \geq 1 - \delta \tag{2.5}$$

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

We now give the following computational conditions for efficient agnostic learning, which are of an ERM-flavor compared to what we have seen before:

1. If $\text{vc}(H_n) \leq \text{poly}(n)$ and there is a poly-time algorithm implementing ERM for $\{H_n\}_n \implies \{H_n\}_n$ is efficiently-agnostically-properly-PAC-learnable.
2. If H_n is efficiently-agnostically-properly-PAC-learnable then $\text{AGREEMENT}_{H_n}(S) \in \text{RP}$, where $\text{AGREEMENT}_{H_n}(S, k)$ is the following decision problem:

$$\text{AGREEMENT}_{H_n}(S, k) = 1 \iff \exists\, h \in H_n, L_S(h) \leq 1 - \frac{k}{|S|}$$

3. *(Contrapositive of above.)* If $\text{RP} \neq \text{NP}$ and solving $\text{AGREEMENT}_{H_n}(S)$ is NP-hard $\implies$ the family $\{H_n\}_n$ is *not* efficiently-agnostically properly-PAC-learnable.

You might wonder which hypotheses classes are efficiently properly agnostic learnable:

- *Poly-time functions?* No, not even in the realizable case.
- *Poly-size depth-2 neural networks?* No, not even in the realizable case either.
- *Halfspaces (linear predictors)?* Note that in the realizable case, halfspaces are efficiently PAC learnable but in the agnostic case, AGREEMENT_{H_n} is NP-hard $\implies$ not efficiently-agnostically PAC-learnable.
- *Conjunctions?* No (but efficiently PAC-learnable in the realizable setting as per Theorem 2.3)
- *Unions of segments on the real line?*
  - $X_n = [0, 1], H_n = \{x \mapsto \vee_{i=1}^{n} 1[a_i \leq x \leq b_i] \mid a_i, b_i \in [0, 1]\}$
  - Yes, efficiently properly agnostically PAC learnable.

As a preview of what's to come, Omar notes that one way the community has bypassed these computational hardness results is to design surrogate losses. For example, instead of using the zero-one loss $\ell^{01}(z, y) = 1[yz \leq z]$ (which our classical risk L_D and L_S typically uses), we can define losses which upper bound ℓ^{01}. Examples:

- *Squared Loss*: $\ell(z, y) = (z - y)^2$
- *Hinge Loss*: $\ell(z, y) = \max(0, 1 - yz)$
- *Logistic Loss*: $\ell(z, y) = \log(1 + \exp(-yz))$
- *Exponential Loss*: $\ell(z, y) = \exp(-yz)$

As a closing for this chapter, we consider the following thoughtful question. If neural networks & deep learning are not efficiently PAC-learnable, why do they perform so well, that too with local search procedures such as Stochastic Gradient Descent (SGD)? This should naturally imply that the problems[7] where deep learning is thriving are not those worst case problems which will determine if a hypothesis class is efficiently-PAC-learnable.

[7] "problems" here are broadly referring to the data distributions D.

Chapter 3

Non-Uniform Learning

3.1 Non-Uniform Learning over Countable Hypothesis Classes (Lecture 5)

We now transition from PAC learning to the study of another learning paradigm of *non-uniform learning*. Observe that hitherto, we have placed a uniform prior over H, where we are a priori assuming each $h \in H$ to be equally likely. Now imagine instead if we placed a prior $p : H \to [0, 1]$ where $\sum_{h \in H} p(h) \leq 1$.

Note that this is a general way for us to encode our prior knowledge of this problem. For example, we may want to induce some sort of bias towards "simpler" functions, so $p(h)$ will encode "simplicity". Or we may have a bias towards predictors with shorter explanations, so $p(h)$ will encode "description length". As an aside, these principles are all at a high-level related to Occam's Razor.

Suppose we go with the latter option of having a bias for a shorter description. Furthermore, we will assume that H is countable. We can then define a bit-descriptor function $d : H \to \{0, 1\}^\star$, which is essentially a description language for H that is *prefix-free*, meaning that $\forall h, h' \in H, d(h)$ is not a prefix of $d(h')$. Then we can define our simplicity-biased prior, $p(h) = 2^{-|d(h)|}$, where by Kraft-McMillan's Inequality we have[1] $\sum_{h \in H} p(h) = \sum_{h \in H} 2^{-|d(h)|} \leq 1$.

Now we are ready to consider the following learning rule of *minimum description length* (MDL):

$$\text{MDL}_p(S) = \text{argmax}_{L_S(h)=0} p(h) = \text{argmin}_{L_S(h)=0} |d(h)|$$

Essentially MDL gives us the consistent hypothesis that is most aligned with our prior, or equivalently has the shortest description length possible. We now present the following (realizable) learning guarantee for MDL:

Theorem 3.1 (MDL Learning Guarantee)**.** *For prior p over a countable H s.t. $\sum_{h \in H} p(h) \leq 1$ (e.g. $p(h) = 2^{-|d(h)|}$ for a prefix-free d), any $\delta \in (0, 1), m \in \mathbb{N}$, and any distribution D s.t. $\exists\, h^* \in H$ where $L_D(h^*) = 0$, with probability $\geq 1 - \delta$ over $S \sim D^m$ we have:*

$$L_D(MDL_p(S)) \leq \frac{\ln(\frac{1}{p(h^*)}) + \ln(1/\delta)}{m} = \frac{\ln(2) \cdot |d(h^*)| + \ln(1/\delta)}{m} \tag{3.1}$$

[1] Observe that our assumption that d is a prefix-free language enables us to use this inequality.

The main thing to notice here is that this MDL guarantee tells us that learning happens at an inversely proportional speed to $|d(h^*)|$, or the length of the best predictor h^*.

Proof. For any $h \in H$ where we have $L_D(h) > \epsilon_h := \frac{\ln(1/p(h))+\ln(1/\delta)}{m}$, we will have:

$$\mathbb{P}_{S\sim D^m}[L_S(h) = 0] \leq (1-\epsilon_h)^m \leq \exp(-e_h m) = p(h) \cdot \delta$$

This is using the same arguments in the derivation of our first ever learning guarantee Theorem 1.2.

So now we apply a union bound to get:

$$\mathbb{P}_{S\sim D^m}[\exists h \in H : L_D(h) > \epsilon_h \wedge L_S(h) = 0] \leq \mathbb{P}_{h\in H}[L_S(h) = 0] \leq \sum_{h\in H} p(h)\delta \leq \delta$$

where inverting the probabilities yields the desired form. Finally, observe that this was for any $h \in H$. Now set $h = \text{MDL}_p(S)$ and apply this bound. Because $\exists\ h^* \in H$ s.t. $L_D(h^*) = 0 \implies p(\text{MDL}_p(S)) \geq p(h^*)$ by definition of MDL. Thus we can loosen this upper bound by using $p(h^*)$ instead of $p(h) = p(\text{MDL}_p(S))$. This arrives us to the desired conclusion. □

Note that this is a very strong result, as it applies to countable hypothesis classes. Example classes that are countable include:

- Class of all computable functions.
- Class numberable with $n : H \to \mathbb{N}$ where we use $p(h) = 2^{-n(h)}$

However, the VC dimension of all computable functions is infinite! So doesn't this imply that all computable functions are not PAC-learnable, as their required sample complexity $m(\epsilon, \delta)$ would be infinite[2]? The reason there is no contradiction is that from Theorem 3.1, we can see that the sample complexity for MDL is given by $m(\epsilon, \delta, h)$. While PAC-learnability requires a sample complexity that works *uniformly* (no pun intended) across all $h \in H$, non-uniform-learnability as we will soon see does not have this requirement. Note however that this does not always mean learning is feasible in the non-uniform setting. For example, $|d(h^*)|$ is very big $\implies$ $m(\epsilon, \delta, h)$ would be large as the $p(h^*)$ term would be very small.

We now present the definition of a hypothesis class being non-uniformly-learnable, which is nearly identical to an agnostically-PAC-learnable hypothesis class (Definition 1.9):

Definition 3.2 (Non-uniformly Learnable Hypothesis Class)**.** *A hypothesis class H is non-uniformly-learnable if there exists a learning rule A such that $\forall(\epsilon, \delta) \in (0, 1)^2, \forall h \in H, \exists\ m(\epsilon, \delta, h) \in \mathbb{N}$ where $\forall D$,*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}\{L_D(A(S)) \leq L_D(h) + \epsilon\} \geq 1 - \delta \tag{3.2}$$

From this, we arrive at a corollary for MDL's sample complexity:

[2]See Fundamental Theory of Statistical Learning, Theorem 1.18.

Corollary 3.3 (MDL Sample Complexity Corollary)**.** *For any prior p over a countable H s.t. $\sum_h p(h) \leq 1$ and any $h^* \in H$, with sample complexity:*

$$m(h^*, \epsilon, \delta) = \frac{\ln(\frac{1}{p(h^*)}) + \ln(1/\delta)}{\epsilon}$$

for any distribution D s.t. $L_D(h^*) = 0$, we have:

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

Structural Risk Minimization (SRM). So far we have only discussed learning guarantees in the realizable setting. We now generalize to consider the agnostic setting. Given a prior p over H, it is not too hard to show that with probability $\geq 1 - \delta$ over $S \sim D^m$, we have:

$$\forall h \in H, L_D(h) \leq L_S(h) + \sqrt{\frac{\ln(1/p(h)) + \ln(2/\delta)}{2m}}$$

where $L_S(h)$ is minimized by ERM and $\sqrt{\frac{\ln(1/p(h)) + \ln(2/\delta)}{2m}}$ is minimized by MDL. Because we want to minimize this upper bound, its natural to define the following "structural risk minimization" (SRM) learning rule $\text{SRM}_p(S)$:

$$\text{SRM}_p(S) = \text{argmin}_{h \in H} L_S(h) + \sqrt{\frac{\ln(1/p(h))}{2m}}$$

We can see that SRM is trying to balance fitting the data with matching the prior. Similar to our previous learning guarantee Theorem 3.1 for MDL in the realizable case, can present the following SRM learning guarantee

Theorem 3.4 (SRM Learning Guarantee over Countable Hypothesis Class)**.** *For prior p over a countable H s.t. $\sum_{h \in H} p(h) \leq 1$, any distribution D, any $\delta \in (0, 1), m \in \mathbb{N}$, with probability $\geq 1 - \delta$ over $S \sim D^m$ we have:*

$$L_D(SRM_p(S)) \leq \inf_{h \in H} \left(L_D(h) + 2\sqrt{\frac{\ln(1/p(h)) + \ln(2/\delta)}{2m}} \right) \tag{3.3}$$

We now aim to generalize beyond our cardinality-based bound to uncountable classes.

3.2 Non-Uniform Learning over Uncountable Hypothesis Classes (Lecture 5)

One such uncountable hypothesis class could be

$$H = \{x \mapsto \text{sign}(f(x)) \mid f : \mathbb{R}^d \to \mathbb{R} \text{ is a polynomial}\}$$

where $\text{vc}(H) = \infty$. Observe that because H is uncountable, there is no valid prior p possible that satisfies $\forall_{h \in H} p(h) > 0$, where we require this condition to ensure that any hypothesis can be learned. But we still want our learning to have an inherent bias towards simpler predictors – for example, what if we could bias towards polynomials in H with lower complexity?

Priors over Hypothesis Classes. Our first attempt is to use priors over entire hypothesis classes, meaning that we will decompose $H = \bigcup_{r \in \mathbb{N}} H_r$, where H_r is the set of degree r polynomial functions. We then can have a prior $p(H_r)$ over these H_r hypothesis classes.

Now just using the VC Dimension bound we know that $\forall r$, choosing δ_r we have:

$$\mathbb{P}_{S \sim D^m}[\forall h \in H_r : L_D(h) \leq L_S(h) + \epsilon_r] \geq 1 - \delta_r, \text{ where } \epsilon_r = O\left(\sqrt{\frac{\text{vc}(H_r) + \ln(1/\delta_r)}{m}}\right)$$

Then setting $\delta_r = p(H_r) \cdot \delta$ and taking a union bound over all r hypothesis classes[3], we will have:

$$\boxed{\mathbb{P}_{S \sim D^m}[\forall r, \forall h \in H_r : L_D(h) \leq L_S(h) + \epsilon_r] \geq 1 - \delta, \text{ where } \epsilon_r = O\left(\sqrt{\frac{\text{vc}(H_r) + \ln(1/p(H_r)) + \ln(1/\delta)}{m}}\right)}$$

Once again wanting to optimize this upper bound ϵ_r, we can arrive at the following form for an SRM-style learning rule:

$$\text{SRM}_p(S) = \text{argmin}_{r, h \in H_r} L_S(h) + c\sqrt{\frac{\text{vc}(H_r) + \ln(1/p(H_r))}{m}}$$

Note that when $H_r = \{h_r\} \implies \text{vc}(H_r) = 0$, and so this would reduce to the standard SRM guarantee over a countable class, up to constants. Conversely, when all prior mass is concentrated on a give hypothesis class (i.e. $p(H_{r_0}) = 1$), then this would simply reduce to our ERM bound over H_r (Corollary 1.16). Thus, using our decomposition of H into countably many hypothesis classes with prior $p(H_r) = 2^{-r}$, we can arrive at the following non-uniformly-learnable sample complexity:

$$m(\epsilon, \delta, h) = O\left(\frac{d^{\deg(h)} + \deg(h) + \ln(1/\delta)}{\epsilon^2}\right)$$

To see how we got this result, see the above boxed result and note that $r = \deg(h)$ and thus we simplify $\text{vc}(H_r)$ as $d^{\deg(h)}$.

We now shift to any general hypothesis class H that we can decompose into $\{H_r\}_{r \in \mathbb{N}}$. Generally speaking, in practice we will have have $\text{vc}(H_r)$ be monotonically increasing whereas $\ln(1/p(H_r))$ is monotonically decreasing. Thus, SRM can be viewed as a bi-criterion optimization problem of minimizing $L_S(h) + r(h)$, where $r(h) = \min\{r \in \mathbb{N} : h \in H_r\}$. Omar notes that we can add a $\lambda \in \mathbb{R}$ parameter in front of $r(h)$ to control SRM's regularization and trajectory (i.e. there is a λ-controlled tradeoff between minimizing $r(h)$ and $L_s(h)$.) Furthermore, we can present the following general theorem (which we will not prove):

Theorem 3.5 (Non-Uniformly-Learnable $\iff$ H is a Countable Union of Finite VC Hypothesis Classes)**.** *A hypothesis class H is non-uniformly-learnable $\iff$ H is a countable union of finite VC classes, or $H = \bigcup_{r \in \mathbb{N}} H_r$ where $\forall r, vc(H_r) < \infty$.*

Before concluding this section, we present a relaxed version of non-uniformly-learnability by allowing the sample complexity to change based on the data distribution D:

[3]The way to see this is that $\mathbb{P}_{S \sim D^m}[\exists\, r, h \in H_r \text{ s.t. } L_D(h) > L_S(h) + \epsilon_r] \leq \sum_r \delta_r = \delta \sum_r p(H_r) \leq \delta$.

Definition 3.6 (Consistently-Learnable Hypothesis Class)**.** *A hypothesis class H is* consistently-learnable *if there exists a learning rule A such that $\forall(\epsilon,\delta)\in(0,1)^2, \forall h\in H, \forall D, \exists\ m(\epsilon,\delta,h,D)\in\mathbb{N}$ where:*

$$\mathbb{P}_{S\sim D^{m(\epsilon,\delta)}}\{L_D(A(S))\leq L_D(h)+\epsilon\}\geq 1-\delta \tag{3.4}$$

Note that this definition is not trivial, meaning that there exists domain X and a hypothesis class H that is consistently-learnable but not non-uniformly-learnable. We now shift to discussing another famous approach for non-uniform learning on uncountable hypothesis classes.

PAC-Bayes Theory. As opposed to placing a prior over hypothesis classes, another approach is to place a prior (not necessarily discrete) over H itself. This is known as *PAC-Bayes*.

We start by defining a randomized/average predictor h_Q (i.e. h is sampled from distribution Q) as:

$$h_Q(x)=y \text{ w.p. } \mathbb{P}_{h\sim Q}[h(x)\neq y]$$

Where we define the expected error of this random predictor h_Q as:

$$L_D(h_Q):=\mathbb{E}_{(x,y)\sim D}[\mathbb{E}_{h\sim Q}1_{[h(x)\neq y]}]$$

With this new technology and idea of hypotheses sampled from a distribution, we present the famous PAC-Bayes generalization error bound:

Theorem 3.7 (PAC-Bayes Generalization Error Bound)**.** *For any class H and any prior distribution P over H, any distribution D over $X\times Y$, any $\delta\in(0,1)$, $m\in\mathbb{N}$ with probability $\geq 1-\delta$ over $S\sim D^m$:*

$$\forall \text{ posterior distributions } Q \text{ over } H: |L_D(h_Q)-L_S(h_Q)|\leq\sqrt{\frac{KL(Q\ ||\ P)+\log(2m/\delta)}{2(m-1)}} \tag{3.5}$$

First observe that this bound is only non-vacuous when $\text{supp}(Q)\subseteq\text{supp}(P)$, as otherwise the $\text{KL}(Q\ ||\ P)$ term would blow up. To emphasize the generality of priors P allowed, observe:

- Suppose we have finite H with $P\sim\text{Unif}(H)$
  - For Q being a point mass on some $h\in H, \text{KL}(Q\ ||\ P)=\log|H|$, as would be expected (Theorem 1.4).
- For discrete distributions P and point mass distributions Q, $\text{KL}(Q\ ||\ P)=\log(1/P(h))$, which matches our MDL bounds Theorem 3.1
- For continuous P (e.g. over linear predictors or polynomials), if Q is a point mass then $\text{KL}(Q\ ||\ P)$ is infinite.

Finally, Omar finishes by noting that the given upper bound on the generalization error in the PAC-Bayes bound Theorem 3.7 alludes to yet another SRM-style algorithm to minimize $L_S(h_Q)+$

$\lambda \mathrm{KL}(Q \mid\mid P)$. It has actually been found that the solution to this minimization problem for any fixed λ is $Q_\lambda(h) \propto P(h) \exp(-\eta L_S(h))$ for some "inverse temperature" η.

That concludes our discussion of non-uniform learning. The main takeaway here is that target concepts are learnable with a sample complexity appropriate for their complexity, description length, etc.

Chapter 4

Boosting & Sample Compression

4.1 Introduction to Boosting (Lecture 6)

What is boosting. We start understanding boosting by looking at what it means to boost δ in the PAC-learning setting, or the confidence parameter of the outputted predictor. The question we ask is if given an algorithm A that PAC-learns a hypothesis class H with $\leq \epsilon$ error and $\geq 1 - \delta_0$ confidence, is it possible for us to construct another algorithm B that (ϵ, δ)-PAC-learns-H for any δ?

The answer to this question is yes[1]! Now what about for boosting the error ϵ? Suppose our learning algorithm A gives us some hypothesis only slightly better than $1/2$ (i.e. $\epsilon_0 = 1/2 - \gamma$ for some small positive γ). Can we construct some other learning algorithm to achieve arbitrarily lower error? Omar notes that this question to boost the error has a rich history since 1988, and has led to the famous AdaBoost algorithm. Before going further in our study of *how* to create this "boosted" classifier, we should be on the same page about what it means to have a "weak learner" that's slightly better than guessing. Note that we are only interested in weak-learners that do *better* than 50% error, as we could achieve a classifier with 50% error that just flips a coin and outputs $+1$ and -1 for heads and tails respectively.

Definition 4.1 (Weak-PAC-learnable Hypothesis Class)**.** *A hypothesis class H is weakly-PAC-learnable if there exists an error parameter $\epsilon_0 = 0.5 - \gamma$ and a learning algorithm A such that $\forall \delta \in (0,1), \forall D$ s.t. $\inf_{h \in H} L_D(h) = 0$,*

$$\mathbb{P}_{S \sim D^{m_A(\delta)}}[L_D(A(S)) \leq \frac{1}{2} - \gamma] \geq 1 - \delta \tag{4.1}$$

We typically refer to γ as the "edge" of our weak learner.

Note that this is in contrast to our familiar definition of (strong) PAC-learning given by Definition 1.8, where we demanded that our learning algorithm be capable of outputting a predict with arbitrarily low error (i.e. ϵ could be anything in $(0,1)$. The major question of today then is:

If a class H is weakly-PAC-learnable, is it also strongly PAC-learnable?

[1] We will see this in Problem Set 2 in the realizable setting.

Omar notes that our guess should be yes, as otherwise we would not be having this lecture. Moreover, there's a simple proof using our Fundamental Theorem of Statistical Learning (Theorem 1.18):

Proof. If $\text{vc}(H) = \infty \implies H$ is not weakly-PAC-learnable. This is because we can always create a realizable distribution D where $A(S)$ will give a hypothesis consistent on the sample $m_A(\delta)$ but not on the remainder of unseen points. In other words, $\forall \gamma > 0$ we can create a large enough distribution D which is realizable (by definition of $\text{vc}(H) = \infty$) where $A(S)$ will make $> 0.5 - \gamma$ error on all the points outside of the small sample so $L_D(A(S)) > 0.5 - \gamma$. Thus using contrapositive we get that if H is weakly-PAC-learnable $\implies \text{vc}(H) < \infty \implies H$ is strongly-PAC-learnable. □

But this is not a terribly satisfying as it does not give us (i) a boosting algorithm B using some weak-learner A (ii) any information on the computational efficiency of B relative to A.

AdaBoost Algorithm. We now present the AdaBoost (adaptive boosting) algorithm, which is a canonical boosting algorithm.

(i) Input: Dataset $S = \{(x_i, y_i)\}_{i=1}^m$, Weak-Learner A, and number of iterations T

(ii) Create a uniform distribution $D_1(i) = 1/m$ over the training examples.

(iii) Now for each iteration $t \in [1, T]$:

- Run weak-learner A on distribution D_t, and set $h_t := A(D_t)$
- Computed the weighted-error ϵ_t:

$$\epsilon_t = L_{D_t}(h_t) = \sum_{i=1}^m D_t(i) 1\{h_t(x_i) \neq y_i\}$$

  By the weak-learning assumption, we have $\epsilon_t < 0.5$.
- Choose $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$
- (*Adaptive* Step). For sample $i \in 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i, \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i, \end{cases} = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

  where our normalization constant is given by

$$Z_t = \sum_{j=1}^m D_t(j) \exp(-\alpha_t y_j h_t(x_j)).$$

(iv) Output the final predictor as a weighted-average over all the predictors we have learned:

$$h_{1:T}(x) = \text{sign}(\sum_{i=1}^T \alpha_t h_t(x))$$

It should be clear on a first read why this algorithm is called "adaptive", in many different ways. Note that the main secret sauce of the algorithm is in the red step: the new distribution $D_{t+1}(i)$ shifts the weights so the next iteration will focus on the mistakes h_t made in the current iteration:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \text{ if } h_t(x_i) = y_i, \text{ decrease sample weight} \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \text{ if } h_t(x_i) \neq y_i, \text{ increase sample weight} \end{cases}$$

We now provide some further technical analysis of why AdaBoost works. Understand first that the weight update can be simplified to:

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i)}{2(1-\epsilon_t)} & \text{if } h_t(x_i) = y_i \\ \frac{D_t(i)}{2\epsilon_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

as the normalizing constant Z_t is:

$$\begin{aligned} Z_t &= \sum_{i:h_t(x_i)\neq y_i} D_t(i)\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + \sum_{i:h_t(x_i)=y_i} D_t(i)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \\ &= \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} = 2\sqrt{\epsilon_t(1-\epsilon_t)}. \end{aligned}$$

So then the error w.r.t. D_{t+1} of the previous iteration's weak learner h_t can be given by:

$$L_{D_{t+1}}(h_t) = \sum_{i:h_t(x_i)\neq y_i} D_{t+1}(i) = \sum_{i:h_t(x_i)\neq y_i} D_t(i)\frac{1}{2\epsilon_t} = \epsilon_t \cdot \frac{1}{2\epsilon_t} = \frac{1}{2}.$$

This prevents the case in which our model at each iteration is returning the same hypothesis, as by definition of weak learner $L_{D_{t+1}}(A(D_{t+1})) \leq 0.5 - \gamma \neq 0.5 = L_{D_{t+1}}(h_t)$.

We now present our first learning guarantee of AdaBoost:

Theorem 4.2 (Upper bound of AdaBoost Training Error w.r.t D_1)**.** *Let $\gamma_t = 1/2 - \epsilon_t$ and let D_1 be any initial distribution over $\{(x_i, y_i)\}_{i=1}^m$. Then, running AdaBoost for T rounds produces predictor $h_{1:T}$ with error w.r.t D_1 bounded by:*

$$L_{D_1}(h_{1:T}) = \mathbb{P}_{(x,y)\sim D_1}[h_{1:T}(x) \neq y] \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp(-2\sum_{t=1}^{T} \gamma_t^2) \tag{4.2}$$

First observe the vacuous case for the above theorem is if $\gamma_t = 0$. We now compute some numerics. Suppose that if $\forall t \in 1, \ldots, T, \gamma_t \geq 10\%$ or equivalently $\epsilon_t \leq 40\%$. Then by the above theorem, we have the following guarantee on the final training error:

$$L_{D_1}(h_{1:T}) \leq (\sqrt{1 - 4 \cdot 0.1^2})^T \approx (0.98)^T$$

For $T = 5$, this upper bound ($\leq$ 0.9ish) is not great. But the above result shows us that the training error is decreasing *exponentially* fast as a function of the number of rounds T. Theorem 4.2 also gives us the slightly weaker corollary:

Corollary 4.3 (Weaker Upper Bound of AdaBoost Training Error w.r.t. D_1)**.** *If* $\max_{1\le t\le T}\varepsilon_t \le \frac{1}{2}-\gamma$ *for some* $\gamma>0$*, then the output predictor* $h_{1:T}$ *of AdaBoost has error w.r.t.* D_1 *bounded by*

$$L_{D_1}(h_{1:T}) = \mathbb{P}_{(x,y)\sim D_1}\left[h_{1:T}(x)\neq y\right] \le \prod_{t=1}^{T}\sqrt{1-4\gamma^2} \le \exp(-2\gamma^2 T).$$

To demonstrate why this may be useful, suppose we have some weak learner A with $\epsilon_0 = 1/2-\gamma$, so the assumptions for the above corollary are satisfied. Then $\forall\epsilon>0$, running AdaBoost for $T=\frac{\ln(1/\epsilon)}{2\gamma^2}$ iterations guarantees $L_{D_1}(h_{1:T})<\epsilon$. Moreover for $\epsilon = 0.5/m$, after $T=\frac{\ln(2m)}{2\gamma^2}$ iterations we will have $L_{D_1}(h_{1:T})<0.5/m \implies$ zero training error.

We now prove Theorem 4.2.

Proof. Consider running AdaBoost for T rounds, and define $F(x)=\sum_{t=1}^{T}\alpha_t h_t(x)$. Then:

- By the weight update equation, we have:

$$\begin{aligned} D_{T+1}(i) &= D_1(i)\times\frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1}\times\cdots\times\frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T} \\ &= \frac{D_1(i)\exp(-y_i\sum_{i=1}^{T}\alpha_t h_t(x_i))}{\prod_{t=1}^{T}Z_t} = \frac{D_1(i)\exp(-y_i F(x_i))}{\prod_{t=1}^{T}Z_t} \end{aligned}$$

- Observe that $h_{1:T}(x)=\text{sign}(F(x))$. If $h_{1:T}(x)\neq y \implies y\cdot F(x)\le 0 \implies \exp(-yF(x))\ge 1$. So $1\{h_{1:T}(x)\neq y\}\le\exp(-yF(x))$.

- Then, we can bound the error of $h_{1:T}$ w.r.t. to D_1 as follows:

$$L_{D_1}(h_{1:T}) = \sum_{i=1}^{m}D_1(i)1\{h(x_i)\neq y_i\} \le \sum_{i=1}^{m}D_1(i)\exp(-y(F(x)) = \sum_{i=1}^{m}D_{T+1}(i)\prod_{t=1}^{T}Z_t = \prod_{t=1}^{T}Z_t$$

- All that is left to do is compute $\prod_{t=1}^{T}Z_t$:

$$\prod_{t=1}^{T}Z_t = \prod_{t=1}^{T}2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_{t=1}^{T}2\sqrt{(1/2-\gamma_t)(1/2+\gamma_t)} = \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2}$$

□

Omar notes the following plot from *Boosting: Foundations and Algorithms* shows that AdaBoost really does work (and beats our theoretical expectations):

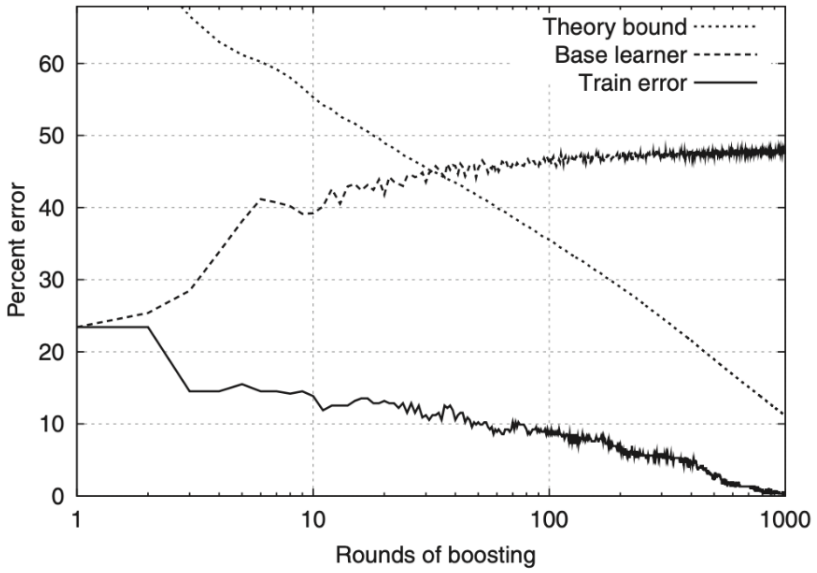


Figure 3.1
The training percent error rate and theoretical upper bound on the training error rate of the combined classifier obtained by using boosting on the entire heart-disease dataset from section 1.2.3. The error rates ϵ_t of the base classifiers on their respective weighted training sets are also plotted.

Note that the error at each time step is computed on the training set w.r.t to D_t, so it makes sense that the base learner (i.e. h_0) would *increase* over time. But this is for the training error, which is not what we really care about. What we really care about is the generalization error of AdaBoost: how can we bound or better understand AdaBoost's generalization error?

Generalization error of AdaBoost. Our first tool we will use is VC Theory. To do so, we'll need a hypothesis class, which we call B:

$$B = \{\text{all hypothesis/predictors outputted by weak-learner } A\}$$

B is essentially our "base class." Now recall the output form of AdaBoost as $h_{1:T} = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$. We can write this as a weighted majority-vote over predictors in B:

$$C_T = \{x \mapsto \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \mid \alpha_{1:T} \in \mathbb{R}^T, h_1, \ldots, h_T \in B\}$$

So our hypothesis class for the AdaBoost output $h_{1:T}$ can be given as C_T (note the meaningful parameterization of C_T by T). So to invoke our standard VC-based uniform convergence bounds, we will want to bound C_T. We have two claims:

- Claim: If B is finite, then $\Gamma_{C_T}(m) \leq (\frac{em}{T})^T |B|^T$. This is because we have $|B|^T$ choices for $h_{1:T}$. Fixing $h_{1:T}$ then gives us a feature map $\phi(x) = (h_1(x), \ldots, h_T(x)) \in \mathbb{R}^T$. Because this has a known VC dimension of $T \implies$ we have $\leq (\frac{em}{T})^T$ possible labelings[2] on this fixed feature map. Thus $\Gamma_{C_T}(m) \leq (\frac{em}{T})^T |B|^T \implies \text{vc}(C_T) = O(T \log |B|)$.

[2]Note that we are playing around with the $\alpha_{1:T} \in \mathbb{R}^T$

- Claim: If B has VC dimension d, then $\Gamma_{C_T}(m) \leq (\frac{em}{T})^T(\frac{em}{d})^{dT}$. This is done with a similar argument as before. Each of the T functions from B we will choose has $(\frac{em}{d})^d$ possible labelings on m points, so there are $\leq (\frac{em}{d})^{dT}$ total possible labelings possible fixing $h_{1:T}$. Then fixing this choice of T hypotheses we can similarly play with the coefficients $\alpha_{1:T}$ to yield a factor of $(\frac{em}{T})^T \implies$ in total we have $\Gamma_{C_T}(m) \leq (\frac{em}{T})^T(\frac{em}{d})^{dT} \implies \text{vc}(C_T) = O(dT \log T)$.

So now that we have good control over $\text{vc}(C_T)$, we are free to invoke our previous VC dimension uniform convergence theorems to arrive at the following:

Theorem 4.4 (AdaBoost Expected Error VC-Dimension Bound)**.** *Running AdaBoost for T rounds using a base class B on a random sample $S \sim D^m$ (where D is an arbitrary distribution) guarantees, with probability at least $1-\delta$, that the output of AdaBoost satisfies*

$$L_D\left(h_{1:T}\right) \leq L_S\left(h_{1:T}\right) + O\left(\sqrt{\frac{vc(B)T\ln(m) + \log(1/\delta)}{m}}\right).$$

Furthermore, with probability at least $1-\delta$, if AdaBoost is consistent with the training data, i.e., $L_S\left(h_{1:T}\right) = 0$, then

$$L_D\left(h_{1:T}\right) \leq O\left(\frac{vc(B)T\ln(m) + \log(1/\delta)}{m}\right).$$

Corollary 4.5 (Weaker VC-Dimension Bound on AdaBoost Expected Error.)**.** *If, in addition to the assumptions above, each base predictor h_t satisfies $L_{D_t}(h_t) = \varepsilon_t \leq \frac{1}{2} - \gamma$ where D_t is a reweighting of S, then setting $T = \ln(2m)/(2\gamma^2)$ guarantees, with probability at least $1-\delta$ over $S \sim D^m$, that*

$$L_D\left(h_{1:T}\right) \leq O\left(\frac{vc(B)(\ln m)^2 + \log(1/\delta)}{\gamma^2 m}\right).$$

There is actually one case which we have conveniently avoided: $\text{vc}(B) = \infty$, or that our weak learner A's hypothesis class has infinite VC dimension. We move onto covering compression based bounds.

4.2 Introduction to Sample Compression Learning (Lecture 6)

We start with a definition.

Definition 4.6 (Sample Compression Schemes)**.** *A learning rule or algorithm $A : (X \times Y)^\star \to Y^X$ admits a compression scheme of size k if there exists a compression map $\kappa : (X \times Y)^\star \to (X \times Y)^k$ such that for any sequence of examples $S = ((x_1, y_1), \ldots, (x_m, y_m))$ it holds that $\kappa(S) \subseteq S$ and $A(S) = A(\kappa(S))$. That is, the output hypothesis $h = A(S)$ can be represented by k examples $\kappa(S)$, and $h = A(\kappa(S))$.*

Essentially we are saying that a learning algorithm A has a compression scheme if it can require only κ-training examples to generate the same output. This immediately gives us the following learning guarantee on the expected error $L_D(A(S))$:

Theorem 4.7 (Sample Compression Schemes $\implies$ Learning)**.** *Let A be a learning rule that admits a compression scheme of size k. Then, for any distribution D, with probability at least $1-\delta$ over $S \sim D^m$,*

$$L_D(A(S)) \leq L_S(A(S)) + \sqrt{\frac{k\ln(m) + \ln(1/\delta)}{2(m-k)}}.$$

Furthermore, with probability at least $1-\delta$ over $S \sim D^m$, if $L_S(A(S)) = 0$, then

$$L_D(A(S)) \leq \frac{k\ln(m) + \ln(1/\delta)}{m-k}.$$

While we will not prove this theorem, Omar notes that it is not too difficult. So how can we use Theorem 4.7 to gain good control over AdaBoost's generalization error? Consider some input S of m examples and a deterministic weak-learner A with $\epsilon_0 = 1/2 - \gamma$ that needs $m_0 \ll m$ samples. At each round of boosting, call A on a sample $S_t \sim D_t^{m_0}$ so $h_t = A(S_t)$, where D_t is the AdaBoost-computed re-weighting of the training dataset S_t. Because each weak hypothesis h_t can be represented by the sequence of m_0 examples it was outputted from, after running for T rounds we would need $\leq Tm_0$ examples to reconstruct weak hypotheses $h_{1:T}$.

This feels a lot like AdaBoost admitting a sample-compression-scheme: we have a specified number of samples Tm_0 we require to get the hypotheses $h_{1:T}$ from AdaBoost(S). More specifically, using all $\bigcup_{i=1}^{T} S_t$ samples, we can recover $h_1, \ldots, h_T$. But there is a catch: AdaBoost(S) includes coefficients $\alpha_{1:t}$, which are computed based on all of S. Thus, AdaBoost$(\bigcup_{i=1}^{T} S_t) \neq$ AdaBoost$(S) \implies$ AdaBoost does not admit a sample-compression-scheme. This motivates the creation of hybrid compression schemes[3]:

Definition 4.8 (Hybrid Compression Schemes)**.** *A learning rule $A : (X \times Y)^\star \to Y^X$ admits a hybrid compression scheme of size k and VC dimension d, if there exists a compression map $\kappa : (X \times Y)^\star \to (X \times Y)^k$ and a hypothesis class map $F : (X \times Y)^k \to \{h : X \to Y\}$ such that for any sequence of examples $S \in (X \times Y)^m$ it holds that $\kappa(S) \subseteq S$, $vc(F(\kappa(S))) \leq d$, and $A(S) \in F(\kappa(S))$. That is, the output hypothesis $h = A(S)$ lies in the hypothesis class $F(\kappa(S))$ which has VC dimension at most d.*

which has the analogous learning guarantee on generalization error:

Corollary 4.9 (Hybrid Compression Schemes $\implies$ Learning)**.** *Let A be a learning rule that admits a hybrid compression scheme of size k and VC dimension d. Then, for any distribution D, with probability at least $1-\delta$ over $S \sim D^m$, if $L_S(A(S)) = 0$, then*

$$L_D(A(S)) \leq O\left(\frac{d\ln\big((m-k)/d\big) + k\ln(m) + \ln(1/\delta)}{m-k}\right).$$

So then we can view AdaBoost as a hybrid compression scheme! The T weak hypotheses $h_{1:T}$ can be presented by a sequence of $k = Tm_0$ examples. Then resulting class from which AdaBoost's output is selected is the set of all weighted majority votes functions:

$$F = \{x \mapsto \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \mid \alpha_{1:T} \in \mathbb{R}^T\}$$

where $\text{vc}(F) = T$. Thus, applying Corollary 4.9 we get:

[3]Omar notes that the creation of this kind of definition is purely for the analysis we are doing now.

Theorem 4.10 (AdaBoost Hybrid Compression Scheme Generalization Error Bound)**.** *Running AdaBoost for T rounds using a deterministic weak learner A that consumes m_0 samples per round guarantees, with probability at least $1-\delta$ over $S \sim D^m$ (for arbitrary distribution D), that if AdaBoost is consistent with the training data, i.e., $L_S(h_{1:T}) = 0$, then*

$$L_D(h_{1:T}) \leq O\left(\frac{T\ln\left(\frac{m-Tm_0}{T}\right) + Tm_0\ln(m) + \ln(1/\delta)}{m - Tm_0}\right) \leq O\left(\frac{Tm_0\ln(m) + \ln(1/\delta)}{m - Tm_0}\right)$$

and similarly applying Corollary 4.5, we get the following corollary:

Corollary 4.11 (AdaBoost Zero Training Error Bound on Generalization Error)**.** *If, in addition to the assumptions above, each base predictor h_t satisfies $L_{D_t}(h_t) = \epsilon_t \leq \frac{1}{2} - \gamma$ where D_t is a reweighting of S, then setting $T = \ln(2m)/(2\gamma^2)$ guarantees with probability at least $1-\delta$ over $S \sim D^m$,*

$$L_D(h_{1:T}) \leq O\left(\frac{m_0(\ln m)^2 + \ln(1/\delta)}{m\gamma^2}\right).$$

We finish with an important theorem connecting weak and strong PAC-learning.

Theorem 4.12 (Weak PAC-Learnable $\iff$ Strong PAC-Learnable)**.** *A hypothesis class H is (efficiently) weakly PAC learnable if and only if is (efficiently) strongly PAC learnable.*

Proof. First observe that the direction strong PAC-learnable $\implies$ weak PAC-learnable is trivial and follows straight from the definitions. We prove now that weak PAC-learning $\implies$ strong PAC-learning, using AdaBoost. Let A be an $(\epsilon_0 = 1/2 - \gamma, \delta/(2T))$-weak-PAC-learner for H (note that even if $\delta = 1/2$, we can "boost" confidence by repeatedly running A and choosing the best hypothesis). So using a union bound over all T hypotheses $h_1, \ldots, h_T$, with probability at least $1 - \delta/2$, all $h_{1:T}$ satisfy $\epsilon_t \leq 1/2 - \gamma$.

Then for any H-realizable distribution D, invoking Corollary 4.11 using $\delta/2$, with probability[4] at least $1-\delta$ over $S \sim D^m$, running AdaBoost on S for $T = \ln(2m)/(2\gamma^2)$ rounds will give us

$$L_D(h_{1:T}) \leq O\left(\frac{m_0(\ln m)^2 + \ln(1/\delta)}{m\gamma^2}\right).$$

which shows strong PAC-learnability as we can make the above RHS arbitrarily small by choosing m appropriately. □

Overfitting Paradox. Recall that $h_{1:T}$ denotes the outputted prediction of AdaBoost. Observe the following behavior and try to think of why it might be confusing:

[4]Note that we are requiring two independent events with $\geq 1 - \delta/2$ probability to occur so we can use the bound from Corollary 4.11. So this bound occurs with $\geq 1 - \delta$ probability.

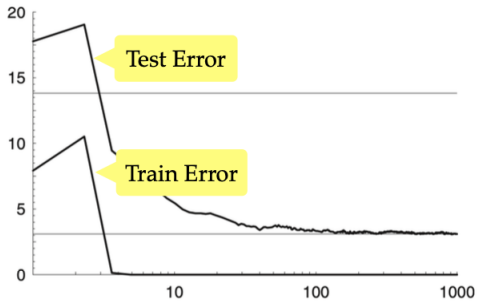


Figure 1.7
The training and test percent error rates obtained using boosting on an OCR dataset with C4.5 as the base learner. The top and bottom curves are test and training error, respectively. The top horizontal line shows the test error rate using just C4.5. The bottom line shows the final test error rate of AdaBoost after 1000 rounds. (Reprinted with permission of the Institute of Mathematical Statistics.)

Even after $L_S(h_{1:T})$ reaches zero, meaning that there is no training error, continuing for more iterations (i.e. $\uparrow T$) leads to the test risk $L_D(h_{1:T})$ *reducing*. This is in contrast to our bound Theorem 4.10, where $L_D(h_{1:T})$ is upper bounded by T. So a natural question then is how do reconcile thse generalization error bounds derived from complexity measures with empirical performance?

4.3 Margin-Based Analysis of Boosting (Lecture 7)

I found reviewing AdaBoost's algorithm before this section very helpful.

Recall AdaBoost's outputted predictor $h_{1:T} = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$. Let us first normalize $h_{1:T}(x)$ by setting each α_t to $\alpha_t / \sum_{t=1}^{T} \alpha_T$. Then given some point $(x, y) \in \mathcal{X} \times \{\pm 1\}$, the *margin* can be given as $y \cdot f(x)$. The most important property here is that $h_{1:T}(x) \neq y \iff y \cdot f(x) \leq 0$. Equipped with this new technology, we provide the following supplement figure to our previous graph of train/test error over T:

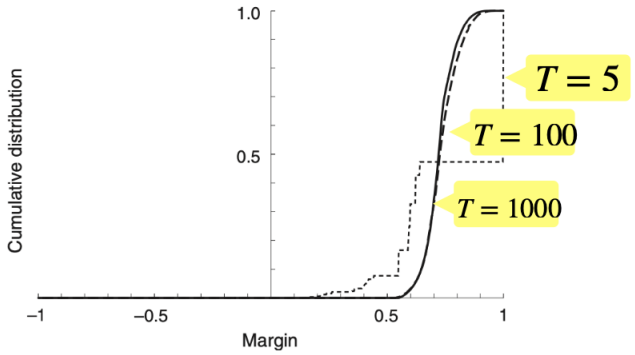


Figure 5.2
The margin distribution graph for boosting C4.5 on the letter dataset showing the cumulative distribution of margins of the training instances after 5, 100, and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden), and solid curves, respectively. (Reprinted with permission of the Institute of Mathematical Statistics.)

This should show us that as T increases, the CDF over the empirical margins not only gets smoother but largely concentrates in the $0.6 - 0.8$ range (which we want!). Note that we required α_t

to be normalized to ensure that the maximum possible margin was either $+1$ or -1, which makes the margins of two different points actually meaningful to compare[5].

We are now ready to consider the following bound on AdaBoost's $\theta-$*margin loss* (before doing so, it might be a good idea to recall what ϵ_t in the AdaBoost algorithm is!):

Theorem 4.13 (AdaBoost θ-margin-loss bound after T iterations)**.** *Let $\gamma_t = \frac{1}{2} - \epsilon_t$, and let D_1 be an arbitrary initial distribution over $\{(x_1, y_1), \ldots, (x_m, y_m)\}$. Then, running AdaBoost for T rounds produces a predictor $f(x) = \sum_{t=1}^{T} a_t h_t(x)$ such that for all $\theta \in [0, 1]$, the θ-margin-loss is bounded as*

$$\Pr_{(x,y)\sim D_1}[y \cdot f(x) \leq \theta] \leq \prod_{t=1}^{T} \sqrt{(1-2\gamma_t)^{1-\theta}(1+2\gamma_t)^{1+\theta}}$$

So the above is a probabilistic bound on any training sample having a margin too-high. Before proceeding further, observe the following properties:

- *Weak-Learning Assumption.* If for all rounds $1 \leq t \leq T, \epsilon_t \leq 1/2 - \gamma$ for some $\gamma > 0$, then the bound can be simplified[6] to $(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}})^T$.
- Moreover, when $\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}} < 1$, the fraction of training examples with a margin $y \cdot f(x) \leq \theta$ will decrease exponentially fast. Through taking logarithms on both sides, we can establish the following statement:

  $$\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}} < 1 \iff \theta < \Upsilon(\gamma) := \frac{-\ln(1-4\gamma^2)}{\ln(\frac{1+2\gamma}{1-2\gamma})}$$

  meaning that $\Upsilon(\gamma)$ gives the maximum possible margin θ such that we can guarantee the percentage of training examples with margin $y \cdot f(x) \leq \theta$ will go to zero with sufficiently large T. Observe for $\gamma \in [0, 1/2]$, we are guaranteed that $\gamma \leq \Upsilon(\gamma) \leq 2\gamma$. Thus, $\forall \theta < \Upsilon(\gamma)$, given enough iterations T, all all margins of our training examples will be $> \theta$ or thus in the limit case $\geq \Upsilon(\gamma) \geq \gamma$ (mathematical justification[7]). Thus, greater edges ($\uparrow \gamma$) yield better guarantees ($\uparrow \Upsilon(\gamma)$) on how large our margins will be post-training.

But this is a bound only on the training examples. We now introduce two things before giving a generalization error bound on the $0-1$ error. Firstly, given some hypothesis class H, let us define convex hull of H as:

$$\text{co}(H) = \left\{ f : x \mapsto \sum_{t=1}^{T} a_t h_t(x) \middle| a_1, \ldots, a_T \geq 0, \sum_{t=1}^{T} a_t = 1, h_1, \ldots, h_T \in H, T \geq 1 \right\}$$

Secondly, observe that for any distribution D, $\mathbb{P}_{(x,y)\sim D}\big[\,\text{sign}(f(x)) \neq y\big] = \mathbb{P}_{(x,y)\sim D}\big[y \cdot f(x) \leq 0\big]$. We now give our nice bound:

[5]For example, without this normalization if we wanted really large margins we could just increase α_t arbitrarily.

[6]Note that this simplification while deceptively simple is actually non-trivial.

[7]For an analytical treatment of this result, pick a sequence $\{\theta_t\}_t$ all strictly bounded above by $\Upsilon(\gamma)$ where $\sup_t \theta_t = \Upsilon(\gamma)$. Then, $\liminf_{T\to\infty} \min_i y_i f_T(x_i) \geq \liminf_{T\to\infty} \theta_T = \Gamma(\gamma) \geq \gamma$.

Theorem 4.14 (Generalization 0-1 Error Bound over co(H))**.** *For any hypothesis class H, any distribution D over $X \times Y$, any $m \in \mathbb{N}$, $\delta \in (0,1)$, and any $\theta \in (0,1)$, with probability at least $1-\delta$ over a sample $S \sim D^m$, for every $f \in \text{co}(H)$,*

$$\mathbb{P}_{(x,y)\sim D}\big[y \cdot f(x) \leq 0\big] \leq \frac{1}{m} \sum_{(x,y)\in S} \mathbf{1}\left[y \cdot f(x) \leq \theta\right] + \frac{2}{\theta}\sqrt{\frac{2\text{vc}(H)\ln\left(\frac{em}{\text{vc}(H)}\right)}{m}} + \sqrt{\frac{2\ln\left(\frac{2}{\delta}\right)}{m}} \tag{4.3}$$

Observe here that we are bounding the $0-1$ error of $\text{sign}(f(\cdot))$ on D through the θ-margin-loss on S and a weird-looking complexity term that scales inversely with θ. Furthermore, this bound is vacuous for $\theta = 0$ and $\theta = 1$ (recall all margins $y \cdot f(x) \leq 1$ because all $f \in \text{co}(H)$ have normalized coefficients α_t.) Omar notes that the proof of this theorem requires using Rademacher Complexity, which we will discuss in a future lecture.

From Theorem 4.14, we get the following relevant corollary to our AdaBoost algorithm:

Corollary 4.15 (AdaBoost Generalization Zero-One Error Bound)**.** *Running AdaBoost using a base class B with edge γ, on $S \sim D^m$ for T rounds, guarantees w.p. $\geq 1-\delta$ that the output of AdaBoost $f = \sum_t \alpha_t h_t$ satisfies*

$$\mathbb{P}_{(x,y)\sim D}\big[y \cdot f(x) \leq 0\big] \leq \frac{1}{m} \sum_{(x,y)\in S} \mathbf{1}\left[y \cdot f(x) \leq \gamma\right] + \frac{2}{\gamma}\sqrt{\frac{2\text{vc}(B)\ln\left(\frac{em}{\text{vc}(B)}\right)}{m}} + \sqrt{\frac{2\ln\left(\frac{2}{\delta}\right)}{m}} \tag{4.4}$$

One thing to observe is that this generalization bound does *not* depend on T, which is unlike other AdaBoost bounds we have seen hitherto.

We now move to proving a very strong statement that gives us a sufficient and necessary condition for weak learnability. First recall as per Theorem 4.13 that if we have a weak-learner with edge γ, for any $\theta < 2\gamma$ we can run a (modified) AdaBoost to guarantee that after a large number of rounds all training examples will have a margin $\geq \theta$. Moreover in the proof of this theorem (which we did not cover), the $\sqrt{(1-2\gamma_t)^{1-\theta}(1+2\gamma_t)^{1+\theta}}$ bound-term at each iteration is given by choosing:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+2\gamma_t}{1-2\gamma_t}\right) - \frac{1}{2}\ln\left(\frac{1+\theta}{1-\theta}\right)$$

and hence why we refer to AdaBoost in this case as a "modified" version. We are now ready to to present our strong theorem on when γ-weak-learnability happens:

Theorem 4.16 (A Necessary and Sufficient Condition for γ-Weak Learnability)**.** *We are given an arbitrary dataset $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and a base hypothesis class H. Then H γ-weakly-learns S if and only if there exists $f \in co(H)$ such that $yf(x) \geq 2\gamma$ for all $(x,y) \in S$.*

Proof. ($\Rightarrow$) If H γ-weakly-learns S, then in principle we can run the modified AdaBoost algorithm (see above choice of α_t) to find a $f \in \text{co}(H)$ with margin θ that is arbitrarily close to 2γ. This is a similar idea to the analytical footnote presented in the proof of Theorem 4.13.

($\Leftarrow$) Suppose there exists $f \in \text{co}(H)$ such that $yf(x) \geq 2\gamma$ for all $(x,y) \in S$. So for *any* distribution D over S, we will have:

$$\mathbb{E}_{(x,y)\sim D}\big[yf(x)\big] = \mathbb{E}_{(x,y)\sim D}\left[y\Big(\sum_t a_t h_t(x)\Big)\right] \geq 2\gamma.$$

Moreover,

$$\mathbb{E}_{(x,y)\sim D}\left[y\Big(\sum_t a_t h_t(x)\Big)\right] = \mathbb{E}_{t\sim\alpha}[\mathbb{E}_{(x,y)\sim D}\left[yh_t(x)\right]]$$

where $t \sim \alpha$ denotes[8] that t is drawn from weights α_t, so higher α_t means a higher chance of drawing that t. This is intuitive because α_t is normalized as $f \in \text{co}(H)$. Since $\mathbb{E}_{t\sim\alpha}[\mathbb{E}_{(x,y)\sim D}\left[yh_t(x)\right]] \geq 2\gamma$, there exists t such that

$$\mathbb{E}_{(x,y)\sim D}\left[yh_t(x)\right] \geq 2\gamma.$$

Finally, observe that because $y, h_t(x) \in \{\pm 1\}$:

$$\mathbb{E}_{(x,y)\sim D}\left[yh_t(x)\right] = \Pr_{(x,y)\sim D}\left[h_t(x) = y\right] - \Pr_{(x,y)\sim D}\left[h_t(x) \neq y\right] = 1 - 2\Pr_{(x,y)\sim D}\left[h_t(x) \neq y\right] \geq 2\gamma$$

so:

$$\Pr_{(x,y)\sim D}\left[h_t(x) \neq y\right] \leq \tfrac{1}{2} - \gamma.$$

which means we have found our $h_t \in \text{co}(H)$ s.t. h_t weakly learns S with edge γ. □

4.4 Sample Compression Schemes, Revisited (Lecture 7)

We start by presenting an analogous definition to Definition 4.6, now for a hypothesis class H as opposed to a learning rule A:

Definition 4.17 (Sample Compression Schemes for a Hypothesis Class)**.** *A hypothesis class H admits a compression scheme of size k if there exists a compression map $\kappa : (X \times Y)^\star \to (X \times Y)^k$ and a reconstruction map $\rho : (X \times Y)^k \to Y^X$ such that for any sequence of labeled examples $S = ((x_1, y_1), \ldots, (x_m, y_m))$ realizable by H (i.e., $\exists h \in H$ with $h(x_i) = y_i$ for all i), the following hold:*

- *$\kappa(S) \subseteq S$;*
- *letting $g = \rho(\kappa(S))$, we have $g(x_i) = y_i$ for all $1 \leq i \leq m$.*

In words, what this means is that we are compressing via κ a dataset S of size m into k examples, from which using the reconstruction map ρ we can (correctly) recover all the labels of all the other $m - k$ examples in S. To some extent, we can think of $\rho(\kappa) : (X \times Y)^\star \to Y^X$ as our new learning rule. As an exercise, Omar leaves us to think about how for hypothesis class $H = \{\text{axis aligned rectangles in } \mathbb{R}^2\}$, there will be a sample compression scheme of $k = 4$ (answer[9]). In parallel to our sample compression scheme learning rule bound Theorem 4.7, we have:

Theorem 4.18 (Compression $\implies$ Learning for a Hypothesis Class)**.** *Let H be a hypothesis class that admits a compression scheme (κ, ρ) of size k. Then, for any distribution D realizable by H, with probability at least $1 - \delta$ over $S \sim D^m$,*

$$L_D(\rho(\kappa(S))) \leq \frac{k\ln(m) + \ln(1/\delta)}{m - k}.$$

[8] This is a slight abuse of notation, but please excuse it!

[9] We only need the four corners of the rectangle which envelopes all $+1$ classified points to understand how all other points in S should be labeled.

Proof. Let D be a distribution realizable by H. Then, for a sample $S \sim D^m$,

$$\begin{aligned}\mathbb{P}_{S\sim D^m}\left[L_D(\rho(\kappa(S))) > \epsilon\right] &= \mathbb{P}_{S\sim D^m}\left[L_S(\rho(\kappa(S))) = 0 \wedge L_D(\rho(\kappa(S))) > \epsilon\right] \\ &\leq \mathbb{P}_{S\sim D^m}\left[\exists S_I := ((x_{i_j}, y_{i_j}))_{j=1}^k \text{ s.t. } L_S\big(\rho(S_I)\big) = 0 \wedge L_D\big(\rho(S_I)\big) > \epsilon\right] \\ &\leq \binom{m}{k}(1-\epsilon)^{m-k} \leq \binom{m}{k} e^{-\epsilon(m-k)}.\end{aligned}$$

as we have $\binom{m}{k}$ possible choices for S_I, where assuming $L_D(\rho(S_I)) > \epsilon \implies$ there is a $\leq (1-\epsilon)^{m-k}$ that all $m-k$ points outside of S_I will be labeled correctly (which is required for $L_S(\rho(S_I)) = 0$). We finish by plugging in:

$$\epsilon = \frac{\ln\binom{m}{k} + \ln(1/\delta)}{m-k}$$

which completes the proof. □

So we have seen that compression $\implies$ learning. But you might be curious at this point if learning $\implies$ compression, which is shown through the following theorem:

Theorem 4.19 (Learning $\implies$ Compression)**.** *Let H be a hypothesis class of VC dimension d. Then H admits a sample compression scheme of size $O(d \log m)$, where m is the size of the sample to be compressed.*

Proof. We construct a compression scheme using boosting.

- Fix a (deterministic) PAC learner A for H (e.g. an ERM for H).
- Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a realizable dataset (by H) to be compressed.
- Run AdaBoost on S using A as the weak learner. Specifically:
  - In each round t, draw a subsample $S_t \sim D_t^{m_0}$, where D_t is an empirical reweighting of the initial uniform distribution D_1 on S, and
  $$m_0 = O\left(\frac{d + \log(\delta_0 = 1/3)}{\epsilon_0 = 1/3}\right) = O(d).$$
  - Use a fixed weight
  $$\alpha = \frac{1}{2}\ln\left(\frac{2/3}{1/3}\right)$$
  to compute the next distribution D_{t+1}. *(Note: using a fixed α implies that the final hypothesis is a simple majority vote over $h_1, \ldots, h_T$).*
- By AdaBoost's guarantee for minimizing the training error on S, the edge in each round satisfies $\gamma \geq \frac{1}{2} - \epsilon_0 = \frac{1}{2} - \frac{1}{3} \geq \frac{1}{6}$. Hence, after
  $$T = \frac{\ln(2m)}{2(1/6)^2} = O(\ln m)$$
  rounds we have
  $$L_S\left(\text{sign}\left(\sum_{t=1}^{T} h_t\right)\right) = 0.$$

- Define the compression map κ to run the above boosting procedure and *select* $T \cdot m_0 = O(d \ln m)$ examples from the m points in S.
- Define the reconstruction map ρ to take those $T \cdot m_0$ examples as input and run the (same) weak learner A to reconstruct $h_1, \ldots, h_T$, and hence the majority-vote classifier.

□

Omar notes that there was a conjecture that any hypothesis class H would admit a sample compression scheme of $O(d)$, as opposed to $O(d \log m)$ as we just showed. In fact, in 2003, computer scientist Manfred Warmuth offered a $600 prize to either prove this conjecture or disprove it with counterexample of a specific H! We conclude this lecture by proving one more result on sample compression schemes for finite VC dimension hypothesis classes:

Theorem 4.20 (Learning $\implies$ Compression II)**.** *Let H be a hypothesis class of VC dimension d. Then H admits a sample compression scheme of size $2^{O(d)}$, where m is the size of the sample to be compressed.*

Proof Sketch. We use the same boosting procedure as before. The main new insights are:

- We use a (proper) PAC-learner A for H (e.g. a particular ERM for H). [*Note: proper means the output of A lies in H.*]
- We run AdaBoost for many rounds T until the following holds:

$$\forall (x, y) \in S : y \cdot \sum_{t=1}^{T} \alpha_t h_t(x) \geq \gamma = \tfrac{1}{6}.$$

  We know this is possible by Theorem 4.13.

- Sparsifying the predictor $\sum_{t=1}^{T} \alpha_t h_t$:
  - After normalizing α_t, observe that $\sum_{t=1}^{T} \alpha_t h_t$ can be viewed as a distribution over the class H. We will call this distribution D.
  - The concept of *Dual VC Dimension*:
    - For every $(x, y) \in X \times Y$, define a function $g_{(x,y)} : H \to \{0, 1\}$ where $g_{(x,y)}(h) = \mathbf{1}[h(x) \neq y]$. Let $G = \{g_{(x,y)} \mid (x, y) \in X \times Y\}$.
    - The dual VC dimension of H, denoted $d^\star$, is defined as the VC dimension of G.
  - Using uniform convergence on the dual space G implies that sampling $N = O\left(\frac{d^\star}{\tau^2}\right)$ i.i.d. predictors from distribution D to choose $h_{i_1}, \ldots, h_{i_N}$ guarantees w.h.p that:

$$(x, y) \in X \times Y : \left| \sum_{t=1}^{T} \alpha_t \mathbf{1}[h_t(x) \neq y] - \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}[h_{i_j}(x) \neq y] \right| \leq \tau$$

– Choosing $\tau = 1/12$ with some manipulation of the above equation will yield us:

$$\forall (x, y) \in S : y \cdot \sum_{j=1}^{N} h_{i_j}(x) \geq y \cdot \sum_{t=1}^{T} \alpha_t h_t(x) - \tfrac{1}{12} \geq \tfrac{1}{6} - \tfrac{1}{12} > 0.$$

so the N hypotheses $h_{i_1}, \ldots, h_{i_N}$ will be sufficient to make a consistent predictor on all of S.

- The compression map κ outputs $N \cdot m_0 = O(d^\star d)$ points from the m points in S (recall m_0 from the previous algorithm).
- The reconstruction map ρ receives as input those $N \cdot m_0 = O(d^\star d)$ points and uses the (same) weak learner A to reconstruct $h_{i_1}, \ldots, h_{i_N}$.
- To conclude, we must show that the dual VC dimension satisfies $d^\star \leq 2^{d+1}$.

□

Chapter 5

Rademacher Complexity

5.1 Introduction to Rademacher Complexity (Lecture 8)

So far we have spoken with learning with respect to the $0-1$ loss. However, what if we are interested in learning with respect to other loss functions (e.g. regression with MSE or classification with logistic loss)? Let us formalize this problem to generalize beyond $\{\pm 1\}$ binary classification problems. Define Z as our domain space (e.g. $X \times Y$) and D to be our distribution over Z. We can let $\mathcal{F} = \{f : X \to \mathbb{R}\}$ be our function class and $\ell : \mathcal{F} \times Z \to \mathbb{R}$ be our loss function. We are interested in functions in F that *compete* with this benchmark:

$$\min_{f \in \mathcal{F}} \mathbb{E}_{z \sim D}[\ell(f, z)]$$

So our main question for today is understanding how to establish learning guarantees in this more general setting. The main tool we will use is Rademacher Complexity, which we define now:

Definition 5.1 (Empirical Rademacher Complexity)**.** *Let Z be a domain and let $\mathcal{F} = \{f : Z \to \mathbb{R}\}$ be a family of functions. For a sequence $S = (z_1, \ldots, z_m) \in Z^m$, the empirical Rademacher complexity of $\mathcal{F}$ with respect to S is*

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_{\sigma_1, \ldots, \sigma_m \sim \text{Unif}\{\pm 1\}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(z_i) \right]$$

Note here that this is an expectation over $\sigma_1, \ldots, \sigma_m \sim \text{Unif}(\{\pm 1\})$. Observe that $\sigma_i f(z_i) > 0$ means the function's output of z_i and the random label σ_i "match" by sign, whereas $\sigma_i f(z_i) < 0$ means the opposite. Thus, the $\sup_{f \in \mathcal{F}}$ term is looking for a function f that matches the random labels $\sigma_1, \ldots, \sigma_m$ as best as possible. Another notable quality is that $\mathcal{R}_S(\mathcal{F})$ does not have an inherent scale. Indeed if $|f|$ is bounded by some $b \in \mathbb{R}$, then $\frac{1}{m} \sum_{i=1}^{m} \sigma_i f(z_i) \leq b$. We now present the main uniform convergence Rademacher Complexity learning guarantee.

Theorem 5.2 (Uniform Convergence by Rademacher Complexity Guarantee)**.** *For any distribution D over Z, function class $\mathcal{F} = \{f : Z \to [-1, +1]\}$, any $m \in \mathbb{N}$, and $\delta \in (0, 1)$, with probability at least $1 - \delta$ over $S \sim D^m$,*

$$\forall f \in \mathcal{F} : \quad \mathbb{E}_{z \sim D}[f(z)] \leq \frac{1}{m} \sum_{i=1}^{m} f(z_i) + 2\mathbb{E}_{S \sim D^m}[\mathcal{R}_S(\mathcal{F})] + \sqrt{\frac{\log(1/\delta)}{2m}}$$

Moreover, with probability at least $1-\delta$ *over* $S \sim D^m$,

$$\forall f \in \mathcal{F}: \quad \mathbb{E}_{z\sim D}[f(z)] \leq \frac{1}{m}\sum_{i=1}^{m} f(z_i) + 2\mathcal{R}_S(\mathcal{F}) + \sqrt{\frac{2\log(2/\delta)}{m}}$$

We should pause to think about why these bounds are remarkable. First, they are both *distribution-dependent* bounds as they take into account the specific distribution D you are working with. The second bound is also *data-dependent*, as the $\mathcal{R}_S(\mathcal{F})$ term depends solely on the S sample you have. This matters because the empirical Rademacher complexity, or its expectation, may differ across various distributions over Z (why?[1]). Thus, both these bounds can beat worst-case VC dimension bounds, which are broad across all data distributions and sample draws. The proof of this uniform convergence result relies on the powerful McDiarmid's Inequality:

Theorem 5.3 (McDiarmid's Inequality)**.** *Let* $Z_1, \ldots, Z_m$ *be independent random variables (not necessarily identically distributed). If* $g: Z^m \to \mathbb{R}$ *satisfies the bounded differences condition: there exist constants* $c_1, \ldots, c_m \in \mathbb{R}$ *such that for all* $z_1, \ldots, z_m \in Z$ *and all* $\tilde{z}_i$,

$$\big|g(z_1, \ldots, z_i, \ldots, z_m) - g(z_1, \ldots, \tilde{z}_i, \ldots, z_m)\big| \leq c_i,$$

then for any $\varepsilon > 0$,

$$\mathbb{P}(g(Z_1, \ldots, Z_m) - \mathbb{E}\big[g(Z_1, \ldots, Z_m)\big] \geq \varepsilon) \leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2}\right).$$

While we will not prove this bound, observe that the Z_i are not assumed to be identically distributed. Note that when each $z_i \in [0,1]$, we can recover Hoeffding's bound using $g(Z_1, \ldots, Z_m) = \frac{1}{m}\sum_{i=1}^m Z_i$. We now are ready to prove the Rademacher Uniform Convergence bound stated in Theorem 5.2.

Proof. This proof will take a lot of clever bounding.

1. Part 1: WTS $\mathbb{E}_{S\sim D^m}[\sup_{f\in\mathcal{F}}(\mathbb{E}_D[f] - \mathbb{E}_S[f])] \leq 2\mathbb{E}_{S\sim D^m}[\mathcal{R}_S(\mathcal{F})]$

$$\begin{aligned}
\mathbb{E}_{S\sim D^m}\big[\sup_f \big(\mathbb{E}_D[f] - \mathbb{E}_S[f]\big)\big] &= \mathbb{E}_{S\sim D^m}\big[\sup_f \big(\mathbb{E}_{S'\sim D^m}\big[\frac{1}{m}\textstyle\sum_{i=1}^m f(z_i')\big] - \frac{1}{m}\sum_{i=1}^m f(z_i)\big)\big] \\
&= \mathbb{E}_{S\sim D^m}\big[\sup_f \mathbb{E}_{S'\sim D^m}\big[\frac{1}{m}\textstyle\sum_{i=1}^m \big(f(z_i') - f(z_i)\big)\big]\big] \\
&\leq \mathbb{E}_{S,S'\sim D^m}\big[\sup_f \frac{1}{m}\textstyle\sum_{i=1}^m \big(f(z_i') - f(z_i)\big)\big] \\
&= \mathbb{E}_\sigma \mathbb{E}_{S,S'\sim D^m}\big[\sup_f \frac{1}{m}\textstyle\sum_{i=1}^m \sigma_i\big(f(z_i') - f(z_i)\big)\big] \\
&\leq \mathbb{E}_\sigma \mathbb{E}_{S'\sim D^m}\big[\sup_f \frac{1}{m}\textstyle\sum_{i=1}^m \sigma_i f(z_i')\big] + \mathbb{E}_\sigma \mathbb{E}_{S\sim D^m}\big[\sup_f \frac{1}{m}\textstyle\sum_{i=1}^m (-\sigma_i) f(z_i)\big] \\
&= 2\mathbb{E}_{S\sim D^m}\big[R_S(\mathcal{F})\big].
\end{aligned}$$

[1] Certain distributions might be easier for $\mathcal{F}$ to fit. For example, a linear regression hypothesis class can more easily D that corresponds to a correlated line compared to random noise over $[-1,1]^2$.

The main tricks here are (1) the use of the ghost sample $S' \sim D^m$ that lets us treat z_i and z_i' symmetrically and (2) the fact that σ_i and $-\sigma_i$ under the $\mathbb{E}_\sigma$ are identical.

2. Part 2: Show $\sup_{f\in\mathcal{F}}(\mathbb{E}_D[f] - \mathbb{E}_S[f])$ concentrates around $\mathbb{E}_{S\sim D^m}[\sup_{f\in\mathcal{F}}(\mathbb{E}_D[f] - \mathbb{E}_S[f])]$

$\sup_{f\in\mathcal{F}}(\mathbb{E}_D[f] - \mathbb{E}_S[f])$ is a random variable as it is a function of $S \sim D^m$. For this step, we will use McDiarmid's Inequality by defining $g(S) = \sup_{f\in\mathcal{F}}(\mathbb{E}_D[f] - \mathbb{E}_S[f])$. We now must show that g satisfies the bounded differences condition. Define $S = (z_1, \ldots, z_m)$ and $\tilde{S} = (z_1, \ldots, \tilde{z}_i, \ldots, z_m)$. Then:

$$\begin{aligned}
|g(S) - g(\tilde{S})| &= \big|\sup_{f\in\mathcal{F}}\big(\mathbb{E}_D[f] - \mathbb{E}_S[f]\big) - \sup_{f'\in\mathcal{F}}\big(\mathbb{E}_D[f'] - \mathbb{E}_{\tilde{S}}[f']\big)\big| \\
&\leq \big|\sup_{f\in\mathcal{F}}\big(\mathbb{E}_D[f] - \mathbb{E}_S[f] - \big(\mathbb{E}_D[f] - \mathbb{E}_{\tilde{S}}[f]\big)\big)\big| \\
&\leq \sup_{f\in\mathcal{F}}\big|\mathbb{E}_{\tilde{S}}[f] - \mathbb{E}_S[f]\big| \\
&= \sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^{m}|f(\tilde{z}_i) - f(z_i)|.
\end{aligned}$$

Note that the last line comes as a fast of $\tilde{S}$ differing by one observation from S. At this point we know $|f| \leq 1 \implies \sup_{f\in\mathcal{F}}|f(\tilde{z}_i) - f(z_i)| \leq 2$, so the bounded differences condition is satisfied. So we can use McDiarmid's Inequality to show the desired concentration, and then use part 1 to recover the first UC bound in the theorem. The second bound in the theorem can be shown by using McDiarmid's Inequality to argue that $\mathcal{R}_S(\mathcal{F})$ concentrates around $\mathbb{E}_{S\sim D^m}[\mathcal{R}_S(\mathcal{F})]$.

□

Unfortunately because our generalization error bound is dependent on the Rademacher Complexity term $\mathcal{R}_S(\mathcal{F})$, we should provide one more bound on $\mathcal{R}_S(\mathcal{F})$. Using *Massart's Finite Lemma*, one can show that for a finite function class $\mathcal{F} = \{f : Z \to \mathbb{R}\}$ we have:

$$\mathcal{R}_S(\mathcal{F}) \leq \left(\max_{f\in\mathcal{F}}\sqrt{\sum_{i=1}^{m}f(x_i)^2}\right)\frac{\sqrt{2\log|F|}}{m}$$

We'll now do something really cool and connect this to VC Dimension. If F is binary, then the above implies that:

$$\mathcal{R}_S(\mathcal{F}) \leq \sqrt{\frac{2\log|\Gamma_S(F)|}{m}}$$

so through use of Sauer's Lemma when F has the finite VC dimension d:

$$\mathcal{R}_S(\mathcal{F}) \leq \sqrt{2\frac{d\log(em/d)}{m}}$$

We will now present a few more properties of Rademacher Complexity, which will help us bound $\mathcal{R}_S(\mathcal{F})$ for more complex $\mathcal{F}$.

Theorem 5.4 (Rademacher Complexity Properties)**.** *Let $\mathcal{F} = \{f : Z \to \mathbb{R}\}$ and $\mathcal{G} = \{g : Z \to \mathbb{R}\}$ be function classes. For any sequence $S \in Z^m$:*

1. *If $\mathcal{F} \subseteq \mathcal{G}$, then $R_S(\mathcal{F}) \leq R_S(\mathcal{G})$.*
2. *For any fixed $h : Z \to \mathbb{R}$, $R_S(h + \mathcal{F}) = R_S(\mathcal{F})$.*
3. *$R_S(\text{co}(\mathcal{F})) = R_S(\mathcal{F})$, where $\text{co}(\mathcal{F}) = \{x \mapsto \mathbb{E}_{f\sim P}[f(x)] \mid P \text{ is a distribution over } \mathcal{F}\}$ is the convex hull of $\mathcal{F}$.*
4. *$R_S(\mathcal{F} + \mathcal{G}) = R_S(\mathcal{F}) + R_S(\mathcal{G})$.*
5. *If $\phi : \mathbb{R} \to \mathbb{R}$ is L_ϕ-Lipschitz, then $R_S(\phi \circ \mathcal{F}) \leq L_\phi R_S(\mathcal{F})$.*

We prove only property (3), which we will use later on.

Proof.

$$R_S(\text{co}(\mathcal{F})) = \mathbb{E}_\sigma[\sup_P \frac{1}{m}\sum_{i=1}^m \sigma_i \mathbb{E}_{f\sim P}[f(z_i)]] = \mathbb{E}_\sigma[\sup_P \mathbb{E}_{f\sim P}[\frac{1}{m}\sum_{i=1}^m \sigma_i f(z_i)]] = \mathbb{E}_\sigma[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^m \sigma_i f(z_i)] = \mathcal{R}_S(\mathcal{F})$$

The main trick here is that $\sup_P \mathbb{E}_{f\sim P} = \sup_{f\in\mathcal{F}}$ as the distribution P over F that yields the largest expected value will just put all its probability mass on the maximal $f \in \mathcal{F}$. □

5.2 Applications of Rademacher Complexity & Other Complexity Measures (Lecture 8)

Now recall our generalization zero-one error bound Theorem 4.14 that we utilized to yield a generalization zero-one error bound for AdaBoost. We are now ready to prove this result with Rademacher Complexity.

Proof. First let us define the function $G : \{(x, y) \mapsto yf(x) \mid f \in \text{co}(H)\}$. Now for any $\theta > 0$, we can consider the following "ramp" loss function ϕ:

$$\phi(u) = \begin{cases} 1 & \text{if } u \leq 0 \\ 1 - u/\theta & \text{if } 0 \leq u \leq \theta \\ 0 & \text{if } u \geq \theta \end{cases}$$

Observe that this is a Lipschitz continuous function with $L_\phi = 1/\theta$. So now we can invoke the Rademacher Complexity uniform convergence bound for the class $\phi \circ G$. With probability $\geq 1 - \delta$ over $S \sim D^m$:

$$\forall f \in \text{co}(\mathcal{H}) : \ \mathbb{E}_{z\sim D}\phi(yf(x)) \leq \frac{1}{m}\sum_{i=1}^m \phi(y_i f(x_i)) + 2R_S(\phi \circ \mathcal{G}) + \sqrt{\frac{2\log(2/\delta)}{m}}$$

Note that by the property (5) of Rademacher Complexity, $\mathcal{R}_S(\phi \circ G) \leq L_\phi \mathcal{R}_S(G)$. Moreover, $\mathcal{R}_S(G) = \mathcal{R}_S(\text{co}(H))$ is clear through definition of empirical Rademacher complexity and by property (3), $\mathcal{R}_S(\text{co}(H)) = \mathcal{R}_S(H)$. Using our previous understanding of $\mathcal{R}_S(H)$ from VC Dimension, we have:

$$\mathcal{R}_S(\phi \circ G) \leq \frac{1}{\theta} \cdot \sqrt{\frac{2\text{vc}(H)\ln(\frac{em}{\text{vc(H)}})}{m}}$$

Thus, we have so far shown the following bound with $\geq 1-\delta$ probability:

$$\forall f \in \text{co}(\mathcal{H}) : \ \mathbb{E}_{z\sim D}\phi(yf(x)) \leq \frac{1}{m}\sum_{i=1}^{m}\phi(y_i f(x_i)) + \frac{2}{\theta}\cdot\sqrt{\frac{2\text{vc}(H)\ln(\frac{em}{\text{vc(H)}})}{m}} + \sqrt{\frac{2\log(2/\delta)}{m}}$$

which is not quite the bound that we want. Observe that $\mathbf{1}\{u \leq 0\} \leq \phi(u) \leq \mathbf{1}\{u \leq \theta\}$ and so $\mathbb{P}_{(x,y)\sim D}[y \cdot f(x) \leq 0] \leq \mathbb{E}_{(x,y)\sim D}\phi(yf(x))$ and moreover $\frac{1}{m}\sum_{i=1}^{m}\phi(y_i f(x_i)) \leq \frac{1}{m}\sum_{i=1}^{m}\mathbf{1}\{y_i \cdot f(x_i) \leq \theta\}$. Thus, we are finished. □

We move onto computing the Rademacher Complexity of some notable hypothesis classes.

Rademacher Complexity of Linear Predictors. Consider $X = \mathbb{R}^d$ and hypothesis class $H_B = \{x \mapsto \langle w, x\rangle \mid w \in \mathbb{R}^d, \|w\|_2 \leq B\}$, where we are putting some bound B on the L^2-norm of w. We can then bound $\mathcal{R}_S(H_B)$ as:

$$\mathcal{R}_S(H_B) = \mathbb{E}_\sigma\Big[\sup_{\|w\|\leq B}\frac{1}{m}\sum_i \sigma_i\langle w, x_i\rangle\Big] = \frac{1}{m}\mathbb{E}_\sigma\Big[\sup_{\|w\|\leq B}\langle w, \sum_i \sigma_i x_i\rangle\Big] = \frac{1}{m}\mathbb{E}_\sigma\Big[B\big\|\sum_i \sigma_i x_i\big\|\Big]$$

$$\leq \frac{B}{m}\sqrt{\mathbb{E}\Big[\big\|\sum_i \sigma_i x_i\big\|^2\Big]} = \frac{B}{m}\sqrt{\sum_i \mathbb{E}[\sigma_i^2]\|x_i\|^2 + \sum_{i\neq j}\mathbb{E}[\sigma_i\sigma_j]\langle x_i, x_j\rangle} = \frac{B}{m}\sqrt{\sum_i \|x_i\|^2} = \sqrt{\frac{B^2}{m}\cdot\frac{1}{m}\sum_i\|x_i\|^2}.$$

where the equality that gets rid of the $\sup_{\|w\|\leq B}$ comes from this general fact of dual norms:

$$\|\sum_i \sigma_i x_i\|_* = \sup\{\langle w, \sum_i \sigma_i x_i\rangle : \|w\|_p \leq 1\}$$

where $\|\cdot\|_*$ indicates the dual norm associated with a given L^p norm $\|w\|_p$. In our case of $p = 2$, it is known the dual norm is the L^2 norm. The first inequality is from Jensen's Inequality as $\sqrt{\cdot}$ is a concave function. Note that this is an example of a scale-sensitive bound, which is in contrast to VC-style bounds which are scale insensitive to our hypothesis classes.

We can change this setup slightly to now look at sparse linear predictors, or more specifically $H_B = \{x \mapsto \langle w, x\rangle \mid w \in \mathbb{R}^d, \|w\|_1 \leq B\}$. We can bound the Rademacher Complexity using similar logic:

$$R_S(H_B) = \mathbb{E}_\sigma\Big[\sup_{\|w\|_1\leq B}\frac{1}{m}\sum_{i=1}^{m}\sigma_i\langle w, x_i\rangle\Big] = \frac{1}{m}\mathbb{E}_\sigma\Big[\sup_{\|w\|_1\leq B}\langle w, \sum_{i=1}^{m}\sigma_i x_i\rangle\Big]$$

$$= \frac{1}{m}\mathbb{E}_\sigma\Big[B\big\|\sum_{i=1}^{m}\sigma_i x_i\big\|_\infty\Big] = \frac{B}{m}\mathbb{E}_\sigma\Big[\max_{1\leq j\leq d}\big|\sum_{i=1}^{m}\sigma_i x_{i,j}\big|\Big]$$

Now for each $j \in [d]$, let us define $v_j := (x_{1,j}, \ldots, x_{m,j}) \in \mathbb{R}^m$, or essentially the slice of S along the jth dimension. Note that $\|v_j\|_2 \leq \sqrt{m} \max_i \|x_i\|_\infty$. Now let us define $V := \{v_1, \ldots, v_d, -v_1, \ldots, -v_d\}$. Observe that[2]:

$$\frac{B}{m}\mathbb{E}_\sigma\Big[\max_{1\leq j\leq d}\big|\sum_{i=1}^m \sigma_i x_{i,j}\big|\Big] = B \cdot \mathcal{R}_S(V) \leq B \cdot \max_i \|x_i\|_\infty \sqrt{\frac{2\log(2d)}{m}}$$

where the last inequality is from Massart's Finite Lemma.

Other Complexity Measures. Omar comments that in learning settings where the scale matters, Rademacher Complexity will shine. We briefly take a look at other complexity measures, starting with VC-Subgraph Dimension or Pollard's[3] pseudo dimension:

Definition 5.5 (Real-Valued Shattering)**.** *Let Z be a domain and $\mathcal{F} = \{f : Z \to \mathbb{R}\}$ a family of functions. We say that $\mathcal{F}$ shatters a sequence $S = (z_1, \ldots, z_m) \in Z^m$ if there exist $\theta_1, \ldots, \theta_m \in \mathbb{R}$ such that for all $y_1, \ldots, y_m \in \{\pm 1\}$ there exists $f \in \mathcal{F}$ with*

$$\forall i \in \{1, \ldots, m\} : y_i = 1 \iff f(z_i) > \theta_i$$

Definition 5.6 (VC-Subgraph Dimension a.k.a Pollard's pseudo-dimension)**.** *The VC-Subgraph dimension of $\mathcal{F}$, denoted $\mathrm{vc}_{\mathrm{sub}}(\mathcal{F})$, is the largest m such that there exists $S \in Z^m$ that is shattered by $\mathcal{F}$.*

As a remark, I found it a good exercise to check that $\mathrm{vc}_{\mathrm{sub}}(\mathcal{F}) = \mathrm{vc}(\{(z, \theta) \mapsto \mathbf{1}[f(z) \leq \theta] \mid f \in \mathcal{F}\})$. Omar also notes that covering numbers are extremely popular:

Definition 5.7 (Covering Numbers Bound)**.** *Let Z be a domain space and $F = \{f : Z \to \mathbb{R}\}$ a family of functions. Let $S = (z_1, \ldots, z_m) \in Z^m$ be a sequence of points. Consider the projection of F onto S,*

$$\Gamma_F(S) = \{(f(z_1), \ldots, f(z_m)) : \ f \in F\} \subseteq \mathbb{R}^m.$$

We say that a finite subset $C \subseteq \mathbb{R}^m$ is an ϵ-cover of $\Gamma_F(S)$ with respect to a metric ρ on $\mathbb{R}^m$ (e.g., $\ell_1, \ell_2, \ell_\infty$) if for all $f \in \Gamma_F(S)$ there exists $v \in C$ such that $\rho(f, v) \leq \epsilon$. The empirical covering number $\mathcal{N}_\rho(\epsilon, F, m)$ is the size of the smallest ϵ-cover of $\Gamma_F(S)$ with respect to ρ (over a worst-case sequence S).

Theorem 5.8 (Empirical Covering Numbers Bound)**.** *Let Z be a domain space and $F = \{f : Z \to [0, B]\}$ a family of bounded real-valued functions with $\mathrm{vc}_{\mathrm{sub}}(F) = d$. Then, for any $\epsilon > 0$,*

$$\mathcal{N}_\infty(\epsilon, F, m) \leq \sum_{i=1}^d \binom{m}{i} \Big(\frac{B}{\epsilon}\Big)^i,$$

which is less than $\left(\frac{emB}{\epsilon d}\right)^d$ for $m \geq d$.

[2] I found the below very confusing. What helped me was first considering $S = [m]$ and each $v_j \in V$ as a function where $v_j(i) = x_{i,j}$. Then apply directly the definition of Rademacher complexity.

[3] Dr. David Pollard from Yale!

Chapter 6

Online Learning

6.1 Introduction to Online Learning (Lecture 9)

So far in this course we have assumed that all our data come from some distribution D over $X \times Y$. Moreover, given some sample $(X \times Y)^\star$, we aim to create a predictor which will generalize well to other test samples *assumed to be drawn from D.*

We now transition to studying a different framework known as the *online learning*. In this framework, the main difference is that we do not assume that the data is coming from some fixed distribution. This is a big jump from our previous studies – can we hope to say anything without this distribution assumption?

First note that we can view this as a game between a learner A and adversary B, where the latter is the one who selects the data to be presented to the former. More precisely, for rounds $t = 1, 2, \ldots$

- Adversary choses an instance $x_t \in X_t$ and sends it to the Learner.
- The Learner predicts a label $\hat{y}_t \in Y$ for x_t.
- The Adversary reveals the correct label $y_t \in Y$ for x_t.

The goal for our learner is to make as few mistakes as possible in this very dynamic setting. Note that if we wanted to write this in math, we can view the Learner as a map:

$$\bigcup_{t\geq 1}(X \times Y)^{t-1} \times X \to Y, \text{ or alternatively } \bigcup_{t\geq 1}(X \times Y)^{t-1} \to Y^X$$

where the predictor $h_t = A((x_1, y_1), \ldots, (x_{t-1}, y_{t-1}))$. We will see soon why we call B an *adversary.*

No Free Lunch in Online Learning. You might wonder if without any more assumptions if it is even possible for us to make a learner algorithm that works. As such, we play the following game where:

$$X = \{\text{students in class}\}, \quad Y = \{\pm 1\}$$

and Omar acts as the adversary and randomly chooses us to predict the points. However, he repeatedly just chooses opposite labels to what we predict! The main idea here is that the adversary can at each step just create a new hypothesis so our (the learner's) predictions are wrong. More rigorously:

Theorem 6.1 (NFL in Online Learning)**.** *For any finite domain $X = \{x_1, \dots, x_N\}$ and any learner A, there exists a target function $f^\star : X \to \{\pm 1\}$ such that A makes N mistakes on the sequence $x_1, x_2, \dots, x_N$.*

Proof. Present the instances $x_1, x_2, \dots, x_N$ to A, and define $f^\star(x_i) = -\hat{y}_i$ where $\hat{y}_i$ is the label predicted by A. □

Corollary 6.2 (Corollary of NFL)**.** *For any infinite domain X and any learner A, there exists a target function $f^\star : X \to \{\pm 1\}$ such that A makes a mistake in each round.*

As such, we can see that our setup so far requires more assumptions. Thus, we will now need some more prior knowledge. To start, we assume $y = f^\star$ for some $f^\star \in H$, and we assume "hypothesis class" $H \subseteq Y^X$ where the learner knows H but not $f^\star$. Moreover, for now we will assume $\{(x_i, y_i)\}_{i \geq 1}$ is realizable by H.

Similar to our previous idea of PAC-learnability, we now present the analogous Mistake Bound Model:

Definition 6.3 (Mistake Bound Model)**.** *Algorithm A learns a hypothesis class H with a mistake bound M if learner A makes at most M mistakes on any sequence of examples consistent with some $f^\star \in H$.*

This is actually a very strong statement as we are making no assumptions on the order of the sequence $\{x_t\}_t$ and asserting that even if we give infinitely many samples to A, A will be correct for all but finitely many samples. The only assumption we make is that $f^\star \in H$.

We now try to understand if *finite* hypothesis classes are learnable under this Mistake Bound Model. The answer is yes, and we give the very simple CONSISTENT$_H$ learner that demonstrates this:

- Initialize the version space $V_1 = H$.
- For rounds $t = 1, 2, \dots$
  - Upon receiving $x_t \in X$, choose a predictor $h_t \in V_t$ and predict $\hat{y}_t = h_t(x_t)$.
  - Upon receiving the true label y_t, update the version space

$$V_{t+1} = \{h \in V_t : h(x_t) = y_t\}.$$

We now prove that this learner has a bounded number of mistakes:

Theorem 6.4 (Mistake Bound for CONSISTENT$_H$ Learner)**.** *On any sequence $((x_t, y_t))_{t \geq 1}$ realizable by H, CONSISTENT$_H$ makes at most $|H| - 1$ mistakes.*

Proof. If $y_t \neq \hat{y}_t$, then h_t is removed from V_t, so $|V_{t+1}| \leq |V_t| - 1$. The true target $f^\star$ always remains in the version space (i.e. $V_1, V_2, \dots$), hence $|V_t| \geq 1$ for all t. Therefore the version space can shrink by at least one only a total of $|V_1| - 1 = |H| - 1$ times, which bounds the total number of mistakes by $|H| - 1$. □

Now can we do better than this $|H| - 1$ mistake bound given by the CONSISTENT$_H$ learner? The answer is yes, as we see with the below HALVING$_H$ learner:

- Initialize the version space $V_1 = H$.
- For rounds $t = 1, 2, \ldots$
  - Upon receiving $x_t \in X$, predict $\hat{y}_t = \text{MAJORITY}\big(h(x_t) : h \in V_t\big)$
  - Upon receiving the true label y_t, update the version space

$$V_{t+1} = \{h \in V_t : h(x_t) = y_t\}.$$

Theorem 6.5 (Mistake Bound for HALVING$_H$ Learner)**.** *On any sequence $((x_t, y_t))_{t\geq 1}$ realizable by H, Halving$_H$ makes at most $\log_2\big(|H|\big)$ mistakes.*

Proof. If $y_t \neq \hat{y}_t$, the majority vote on $\{h(x_t) : h \in V_t\}$ is wrong, so at least half of the hypotheses in V_t misclassify x_t and are eliminated when we update to V_{t+1}. Hence $|V_{t+1}| \leq |V_t|/2$. The true target $f^\star$ always remains in the version space, so $|V_t| \geq 1$ for all t. After m mistakes we therefore have $1 \leq |V_{t+1}| \leq |H|/2^m$, which implies $m \leq \log_2 |H|$. □

This is a sure improvement! Before seeing if we can improve this even further or generalize this to infinite hypothesis classes, we present a some preliminaries under the mistake bound model:

Definition 6.6 (Conservative Online Learning Algorithm)**.** *An online learning algorithm A is* conservative *if it only changes its state (i.e. predictor) when it makes a mistake.*

Theorem 6.7 (Conservative Online Learning Algorithms Preserve Mistake Bound)**.** *If a hypothesis class H is online learnable with a mistake bound M, then it is online learnable by a conservative algorithm with mistake bound M.*

Proof sketch. Given any online learning algorithm A with mistake bound M, define a new algorithm A' that simulates A and, whenever A predicts correctly on (x_t, y_t), discards any function update (i.e. does not change the state). When A errs, A' performs the same update as A. So A' changes its state only on mistakes, and so it is conservative.

Moreover, the state of A' after seeing N examples is exactly the state of A if it only sees the examples on which it made a mistake, which is a legal sequence of examples as per Definition 6.3. Thus if A sees $\geq M$ mistakes out of these N samples, after the Mth mistake it is guaranteed to give a state/predictor that yields no more mistakes $\implies$ A' by the Mth mistake will also no longer give anymore mistakes. So A' makes at most M mistakes. □

We now test the boundaries of our theory with the following *infinite* threshold hypothesis class:

$$X = [0, 1], \quad H = \{x \mapsto \text{sign}(x - \theta) \mid \theta \in [0, 1]\}$$

Recall Corollary 6.2. We now show that H is NOT learnable under the Mistake Bound Model:

Proof. We present the following procedure to show that for any learner A, we can make sure it has infinitely many mistakes.

- Start with $l_1 = 0, r_1 = 1$.

- For rounds $t = 1, 2, \ldots$
  - Present $x_t = l_t + (r_t - l_t)/2$.
  - If A predicts $\hat{y}_t = +1$, set $y_t = -1$ and update $l_{t+1} = x_t$.
  - If A predicts $\hat{y}_t = -1$, set $y_t = +1$ and update $r_{t+1} = x_t$.
- Observe that for all rounds t, any threshold $\theta \in (l_{t+1}, r_{t+1})$ is consistent with $((x_t, y_t))_{t'=1}^{t}$.
- So there is no bound on the number of mistakes our learner A will make on this H-realizable sequence of examples.

□

Observe that the things that is being exploited here is the infinite precision of θ. This is a big (negative) result. First, we should realize that online learnability is a generally stronger condition than PAC-learnability. Secondly, this result on H implies we *cannot* learn halfspaces in higher dimensions! This still begs the question:

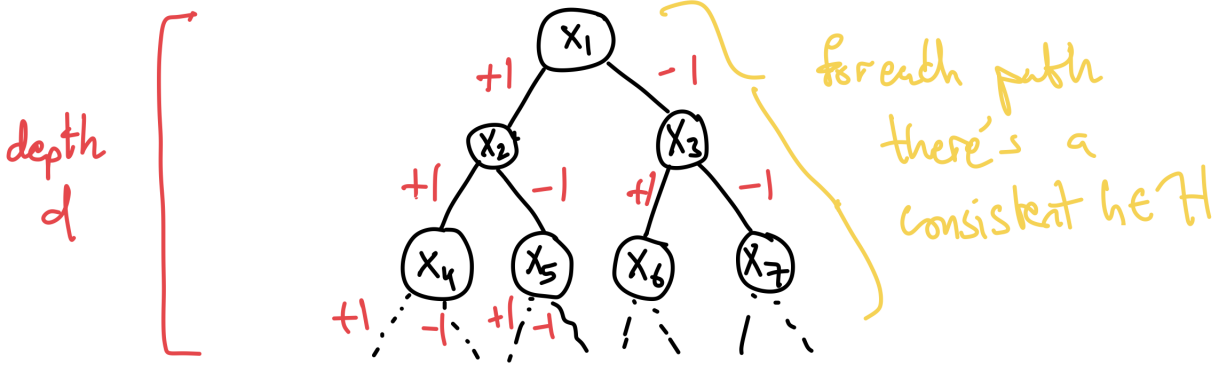
Can we have a characterization of which classes H are online learnable?

We back up a little bit. Recall how in PAC-learning, we had the combinatorial characterization of the VC dimension to easily tell us if something is PAC-learnable. For online learnability, this parallel quantity will be based on Littlestone trees and dimension.

6.2 Littlestone Dimension for Online Learning (Lecture 9)

Definition 6.8 (Littlestone Trees Definition)**.** *A Littlestone tree of depth d is a complete binary tree whose internal nodes are labeled by instances from X, and whose two edges connecting a node to its children are labeled with $+1$ and -1 such that every finite path emanating from the root is consistent with some concept in H. That is, a Littlestone tree is a collection $\{x_u : 0 \le k < d,\ u \in \{\pm 1\}^k\} \subseteq X$ such that for every $y \in \{\pm 1\}^d$, there exists $h \in H$ with $h(x_{y_{1:k}}) = y_{k+1}$ for $0 \le k < d$.*

The definition is a little thick, so we just show a picture:



We now define the *Littlestone Dimension*, which is our magic sauce:

Definition 6.9 (Littlestone Dimension)**.** *The Littlestone dimension of H, denoted lit(H), is defined as the largest integer d such that there exists a Littlestone tree for H of depth d.*

Using this new technology $\text{lit}(H)$, we now demonstrate its relevance to understanding when online learning is possible:

Theorem 6.10 (Lower Littlestone Bound of Mistake Bound of A)**.** *For any class H and any learner A, the mistake bound of A for learning H is $\geq$ lit(H).*

Proof. We first let $T := \text{lit}(H)$ and consider a Littlestone tree for H of depth T. We will start with x_1 being the root of the tree. Now for iteration $1 \leq t \leq T$, we will present the root x_t of the current subtree to the learner A. Now defining A's prediction as $\hat{y}_t$, we will recurse downwards to the opposite subtree which will label x_t with $-\hat{y}_t$. So the A has effectively mislabeled x_t.

Note then by definition of our Littlestone tree with depth T, this path of depth T is realizable by H. So, we have forced our learner A to make $\geq T$ mistakes. □

Theorem 6.11 (Upper Littlestone Bound of Some Learner A)**.** *For any class H, there exists a learner A that learns H with a mistake bound of $\leq$ lit(H).*

Note that this is generally a tighter improvement over the $\log_2(|H|)$ upper bound.

Proof. We first present the standard optimal algorithm (SOA), which we will use our learner. Then we will later show that it has at most $\text{lit}(H)$ mistakes. We start SOA by initializing the version space $V_1 = H$ and then for rounds t proceed as follows:

- Receive $x_t \in X$
- For $r \in \{\pm 1\}$, let $V_t^{(r)} = \{h \in V_t : h(x_t) = r\}$
- Predict $\hat{y}_t = \text{argmax}_{r \in \{\pm 1\}} \text{lit}(V_t^{(r)})$. So we are predicting the label which's associated hypothesis class has the higher Littlestone dimension.
- Upon receiving the true label y_t, update the version space $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$.

First note that because $V_{t+1} \subseteq V_t$ always, $\text{lit}(V_{t+1}) \leq \text{lit}(V_t)$. If $y_t \neq \hat{y}_t$ then we claim this implies that $\text{lit}(V_{t+1}) \leq \text{lit}(V_t) - 1$. First observe that $V_{t+1} = V_t^{(y_t)}$. BWOC, assume $\text{lit}(V_{t+1}) = \text{lit}(V_t^{(y_t)}) = \text{lit}(V_t)$. Then we know for sure by our argmax rule that $\text{lit}(V_t^{(\hat{y}_t)}) \geq \text{lit}(V_t^{(y_t)}) = \text{lit}(V_t)$. And we also know that $V_t^{(\hat{y}_t)} \subseteq V_t \implies \text{lit}(V_t^{(\hat{y}_t)}) \leq \text{lit}(V_t)$. So therefore, this implies:

$$\text{lit}\big(V_t^{(+1)}\big) = \text{lit}\big(V_t^{(-1)}\big) = \text{lit}(V_t)$$

So this means $V_t^{(+1)}$ and $V_t^{(-1)}$ both have associated d-depth Littlestone trees. So we can just create a new tree with x_t at the root with the associated d-depth Littlestone trees on the $+1$ and -1 edge accordingly. But because we are only using functions in V_t to create this bigger tree of depth $\text{lit}(V_t) + 1$, this means the Littlestone dimension of V_t is $\geq \text{lit}(V_t) + 1$, which is a contradiction of the maximality of $\text{lit}(V_t)$. Thus, we can conclude in the case of a mistake at iteration $t, \text{lit}(V_{t+1}) \leq \text{lit}(V_t) - 1 \implies$ the total number of mistakes will be at most $\text{lit}(V_1) = \text{lit}(H)$. □

Corollary 6.12 (Mistake Bound Model Learnability $\iff$ Finite lit(H))**.** *A class H is learnable in the Mistake Bound model if and only if the Littlestone dimension of H, lit(H), is finite.*

From the above corollary, we would be interested in the Littlestone dimensions of our thresholds hypothesis class H. Indeed, $\text{lit}(H) = \infty$ as illustrated by the below figure[1]:

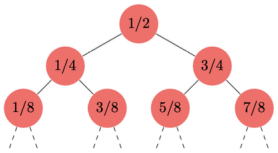


Figure from S. Shalev-Shwartz and S. Ben-David,
Understanding Machine Learning: From Theory to Algorithms

This should be quite impressive considering $\text{vc}(H) = 1$. Similarly, for halfspaces G (linear predictors) in $\mathbb{R}^d$, $\text{lit}(G) = \infty$ but $\text{vc}(G) = d + 1$. Moreover, in general it holds that for any class H, $\text{vc}(H) \leq \text{lit}(H)$ as using any VC shattered set of points we can construct a Littlestone tree with that depth[2].

So far we have only seen finite hypothesis classes be online learnable in part because they all have finite Littlestone dimension. So are there any examples of online learnable infinite hypothesis classes? Indeed the answer is yes:

$$X = [0, 1], H = \{x \mapsto \mathbf{1}[x = \theta] \mid \theta \in [0, 1]\}$$

Proof. We claim that $\text{lit}(H) = 1$. It is trivial that $\text{lit}(H) \geq 1$. Moreover, we can consider a learner A which will predict 0 until it receives a true label of 1, and then memorize this θ and predict the correct answer after that. Then A will make at most 1 mistake and so by Theorem 6.10, $1 \geq \text{lit}(H)$. Thus we are finished. □

Online-to-Batch Conversion Schemes. We end this lecture by understanding the relationship between Mistake Bound Learnability and PAC learnability. To start, one implies the other as:

$$\text{Mistake Bound Learnability} \implies \text{vc}(H) \leq \text{lit}(H) < \infty \implies \text{vc}(H) < \infty \implies \text{PAC-learnability}$$

But to constructively prove this result, we present the *Longest Running Survivor* technique. To start, we take as input $(\epsilon, \delta) \in (0, 1)^2$, a (conservative[3]) online learner A with mistake bound M, and training samples $S = \{(x_i, y_i)\}_{i=1}^m \sim D$. We will then run the online learner A on S until it can produce a hypothesis h that survives $\geq \frac{1}{\epsilon} \ln(M/\delta)$ examples. We claim this leads to a PAC-satisfiable hypothesis:

Theorem 6.13 (Longest Running Survivor Technique (LRST) Yields PAC-Satisfiable Hypothesis)**.** *For any class H, let A be an online learning algorithm with Mistake Bound $M(H)$. Then the Longest Running Survivor technique halts after seeing $O\left(\frac{M \log(M/\delta)}{\epsilon}\right)$ examples and with probability at least $1 - \delta$, produces a hypothesis with error $\leq \epsilon$.*

[1] Note that in our demonstration that thresholds are NOT learnable under the Mistake Bound Model, we essentially demonstrated that we can create a Littlestone Tree of infinite depth.

[2] Just use the same instances across our shattered set for each level. So if $\{x_i\}_{i=1}^m$ is our sequence of m shattered points, we can make a Littlestone tree with instance x_1 as head, instance x_2 for the second level, and so forth.

[3] By Theorem 6.7, we can WLOG assume our learner to be conservative.

Proof. Note that for any produced hypothesis h:

$$\mathbb{P}(\text{any single survived } h \text{ has error} > \epsilon) \leq (1-\epsilon)^{\ln(\delta/M)/\epsilon} \leq \delta/M$$

Furthermore, because A is conservative it will at most have M different hypotheses outputted[4]. So taking a union bound over all of them, there is a less than δ probability any of them will have error $> \epsilon$. □

6.3 Beyond Realizability: Introducing Regret (Lecture 10)

We have so far assumed that the sequence $\{(x_i, y_i)\}_i$ fed into our online learner is realizable by our hypothesis class $H \subseteq Y^X$. We used this assumption in our understanding of the Mistake Bound Model of (online) learning. But now consider the case in which an adversary's inputted examples and reported labels are not consistent with some function in H.

This brings us to this notion of *regret*. We will still assume a sequential game between a Learner and an Adversary, where the Learner has access to a reference/benchmark class H. But we are now longer assuming $\{(x_i, y_i)\}_i$ is realizable by H.

The goal for our learner now is *not* to achieve finitely many mistakes, but instead the weaker goal of minimizing regret:

$$\sum_{t=1}^{T} 1\{\hat{y}_t \neq y_t\} - \min_{h \in H} \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\}$$

So it's okay if our learner screws up, we just want it to screw up less relative to what's possible in H. Note that this also recovers the previous realizable setting (think about what happens to $\min_{h \in H} \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\}$).

As is no surprise, we will try to make a bound for the regret. Naturally, we should shoot for a regret bound that sub-linearly grows with T, such as $\sqrt{T}$, as this would mean the average increase in regret per time step would $\downarrow 0$ as $T \to \infty$. Unfortunately, given our current setup, sublinear regret is unachievable. Consider the following counter example:

- Consider a stupid hypothesis class $H = \{h_+, h_-\}$, where h_+ always predicts $+1$ and h_- is vice versa
- For any (deterministic) learner A, there will always exist a sequence $\{(x_i, y_i)\}_{i=1}^{T}$ such that A makes T mistakes (refer to Theorem 6.1)
- This means that $\min_{h \in H} \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\} \leq T/2 \implies$ the regret of $A \geq T/2$

So to go beyond this result, we have to consider *randomized* learners. What this means is that on each round t, the adversary will decide on a label y_t without observing the randomness the learner might perform to produce a prediction (e.g. flipping a coin). Hence, our previous proof will not apply (in particular the word "always" in the second bullet). Given this randomization in A, we are now going to analyze the *expected* regret of the Learner:

[4]This also means it will take at most $M \cdot \ln(M/\delta)/\epsilon$ examples, as each hypothesis requires $\ln(M/\delta)/\epsilon$.

$$\sum_{t=1}^{T} \mathbb{E}_A[1\{\hat{y}_t \neq y_t\}] - \min_{h \in H} \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\}$$

Now can we achieve sub-linear (expected) regret with randomized learners? The answer is yes. One procedure that demonstrates this is the very famous *multiplicative weights* procedure for finite $|H|$, which we detail below:

- Initialize weights $W_1(h) = 1$ for each $h \in H$ and distribution $P_1(h) = W_1(h)/|H|$.
- Now for rounds $t = 1, 2, \ldots, T$:
  - Adversary chooses $x_t \in X$.
  - Learner randomly draws a predictor $h_t \sim P_t$ and predicts $\hat{y}_t = h_t(x_t)$.
  - Adversary reveals true label y_t.
  - Learner updates distribution

$$W_{t+1}(h) = W_t(h) \exp\left(-\eta \mathbf{1}\{h(x_t) \neq y_t\}\right), \qquad P_{t+1}(h) = \frac{W_{t+1}(h)}{W_{t+1}},$$

    where $W_{t+1} := \sum_{h \in H} W_{t+1}(h)$ is a normalization factor.

So we are essentially always re-weighting in a way to place more mass on the "good" predictors in H. Morally, this is also similar to AdaBoost as we shall see at the end of the lecture. We now present the following bound on the expected regret:

Theorem 6.14 (Multiplicative Weights Sub-Linear Expected Regret Bound)**.** *On any sequence $\{(x_t, y_t)\}_{t=1}^T$ and any $\eta > 0$, the regret of Multiplicative Weights satisfies*

$$\sum_{t=1}^{T} \mathbb{E}\big[\mathbf{1}\{\hat{y}_t \neq y_t\}\big] - \min_{h \in H} \sum_{t=1}^{T} \mathbf{1}\{h(x_t) \neq y_t\} \leq \frac{\ln |H|}{\eta} + \frac{T\eta}{8}.$$

We now prove this statement:

Proof.

Lemma 6.15. *From Hoeffding's Bound, for any random variable X with $a \leq X \leq b$, $\forall s \in \mathbb{R}$,*

$$\ln \mathbb{E}[e^{sX}] \leq s\mathbb{E}[X] + \frac{s^2(b-a)^2}{8}$$

For each $h \in H$, let us define $L_h := \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\}$. Observe then the following (note W_1, W_{T+1} were defined as normalization factors):

$$\ln \frac{W_{T+1}}{W_1} = \ln(\sum_{h \in H} \exp(-\eta L_h)) - \ln |H| \geq \ln(\max_{h \in H} \exp(-\eta L_h)) - \ln |H| = -\eta \min_{h \in H} L_h - \ln |H|, \quad (\star)$$

Next, for each $t = 1, \ldots, T$, by definition of the update,

$$\ln \frac{W_{t+1}}{W_t} = \ln \frac{\sum_{h \in H} W_t(h) \exp\big(-\eta \mathbf{1}\{h(x_t) \neq y_t\}\big)}{\sum_{h \in H} W_t(h)} = \ln \mathbb{E}_{h \sim P_t}\left[\exp(-\eta \mathbf{1}\{h(x_t) \neq y_t\})\right].$$

Invoking our lemma above implies that

$$\ln \frac{W_{t+1}}{W_t} \leq -\eta \mathbb{E}_{h \sim P_t}\left[\mathbf{1}\{h(x_t) \neq y_t\}\right] + \frac{\eta^2}{8}.$$

So we have:

$$\ln \frac{W_{T+1}}{W_1} = \ln \left(\prod_{t=1}^{T} \frac{W_{t+1}}{W_t} \right) = \sum_{t=1}^{T} \ln \frac{W_{t+1}}{W_t} \leq -\eta \sum_{t=1}^{T} \mathbb{E}_{h \sim P_t}\left[\mathbf{1}\{h(x_t) \neq y_t\}\right] + \frac{\eta^2 T}{8}.$$

Putting the above lower bound and previous upper bound (see ($\star$) equation) for $\ln(W_{T+1}/W_1)$ together gives

$$-\eta \min_{h \in H} L_h - \ln |H| \leq \ln \frac{W_{T+1}}{W_1} \leq -\eta \sum_{t=1}^{T} \mathbb{E}_{h \sim P_t}\big[\mathbf{1}\{h(x_t) \neq y_t\}\big] + \frac{\eta^2 T}{8}.$$

By rearranging terms and dividing by η,

$$\sum_{t=1}^{T} \mathbb{E}_{h \sim P_t}\big[\mathbf{1}\{h(x_t) \neq y_t\}\big] - \min_{h \in H} \sum_{t=1}^{T} \mathbf{1}\{h(x_t) \neq y_t\} \leq \frac{\ln |H|}{\eta} + \frac{T\eta}{8}.$$

Finally, observe that $\mathbb{E}_{h \sim P_t}\big[\mathbf{1}\{h(x_t) \neq y_t\}\big] = \mathbb{E}\big[\mathbf{1}\{\hat{y}_t \neq y_t\}\big]$ by the prediction strategy of Multiplicative Weights. So we have proved the desired result. □

Optimizing (i.e. minimizing) this upper bound as a function of η, we have:

Corollary 6.16 (Multiplicative Weights Sub-Linear Regret Bound)**.** *Choosing $\eta = \sqrt{8 \ln |H|/T}$, achieves a sub-linear regret of $\sqrt{(T/2) \ln |H|}$.*

Sub-Linear Regret Bound for Finite Littlestone Dimension. So the multiplicative weights procedure achieves sub-linear expected regret on finite classes H. Let's relax the finiteness assumption to hypothesis classes with finite Littlestone Dimension[5] $\text{lit}(H)$. We can make the following regret guarantee:

Theorem 6.17 (Sub-Linear Expected Regret Bound for $\text{lit}(H) < \infty$.)**.** *For any class H, there exists an online learner A such that on any sequence $\{(x_t, y_t)\}_{t=1}^{T}$, A achieves a regret guarantee of*

$$\sum_{t=1}^{T} \mathbb{E}_A\big[\mathbf{1}\{\hat{y}_t \neq y_t\}\big] - \min_{h \in H} \sum_{t=1}^{T} \mathbf{1}\{h(x_t) \neq y_t\} \leq \sqrt{(T/2)\,\text{lit}(H) \ln\big(eT/\,\text{lit}(H)\big)}.$$

[5]It is a good exercise to think of why finite VC Dimension classes do not tell us anything on if sub-linear regret bounds are possible. **Answer**: Recall our thresholds hypothesis class H with a meager VC dimension of one. It was not learnable in the (realizable) Mistake Bound model setting as at each time step in the Learner-Adversary game we could create a consistent sequence with H where the learner always screwed up. Thus, there is no hope for sub-linear regret as the $\sum_{t=1}^{T} 1\{\hat{y}_t \neq y\}$ can always be made $\propto T$.

We now show how such a learner A is possible. To do so, we will run the Multiplicative Weights algorithm on a suitably defined set of experts[6]. Moreover, we will use different instantiations of the Standard Optimal Algorithm (SOA) to form these experts. Recall that we used the SOA learner in Theorem 6.11 for its mistake bound of $\text{lit}(H)$. Then we will finish by arguing that for all $h \in H$ and each $x_1, \dots, x_T$, we will find some expert that behaves like h on $x_1, \dots, x_T$.

We now go into the details. For each $M \leq \text{lit}(H)$ and indices $1 \leq i_1 < \cdots < i_M \leq T$, we will define an expert:

Expert $(i_1, \dots, i_M)$**.**
Initialize the version space $V_1 = H$.
For rounds $t = 1, 2, \dots, T$:

1. Receive $x_t \in X$.
2. For $r \in \{\pm 1\}$, let $V_t^{(r)} = \{h \in V_t : h(x_t) = r\}$.
3. Let $\tilde{y}_t = \arg\max_{r \in \{\pm 1\}} \text{lit}\big(V_t^{(r)}\big)$.
4. If $t \in \{i_1, \dots, i_M\}$, then predict $\hat{y}_t = 1 - \tilde{y}_t$. Otherwise, predict $\hat{y}_t = \tilde{y}_t$.
5. Update the version space $V_{t+1} = \{h \in V_t : h(x_t) = \hat{y}_t\}$.

where the main difference between this algorithm and SOA is on predictions for indices $i_1, \dots, i_M$ and the version space update. At a high level, an Expert$(i_1, \dots, i_M)$ simulates a game a between SOA and the Adversary, where it assumes a-priori that SOA will make a mistake on rounds $i_1, \dots, i_M$. We are going to run Multiplicative Weights with all these N possible experts where N is given by[7]:

$$N = \sum_{M=0}^{\text{lit}(H)} \binom{T}{M} \leq \left(\frac{eT}{\text{lit}(H)}\right)^{\text{lit(H)}}$$

Using the regret guarantee of Multiplicative Weights, we have:

$$\sum_{t=1}^{T} \mathbb{E}[1\{\hat{y}_t \neq y_t\}] - \min_{(i_1,\dots,i_M)} \sum_{t=1}^{T} 1[\{\text{Expert}(i_1, \dots, i_m)(x_t) \neq y_t\}] \leq \sqrt{(T/2)\ln N}$$

But we are not done as we aim to compete not over the set of experts, but over the entire hypothesis class H. To do this, we will use the following lemma:

Lemma 6.18. *For any class H, any sequence $x_1, \dots, x_T$, and any $h \in H$, there exists $M \leq$ lit(H) and indices $1 \leq i_1, \dots, i_M \leq T$ such that, when running Expert$(i_1, \dots, i_M)$ on the sequence $x_1, \dots, x_T$, the expert predicts the label $h(x_t)$ on each round $1 \leq t \leq T$. In other words, the expert behaves exactly like h on the sequence $x_1, \dots, x_T$.*

Proof. Consider running SOA on the sequence $(x_1, h(x_1)), \dots, (x_T, h(x_T))$. By guarantees of SOA, it makes $\leq \text{lit}(H)$ mistakes. Call the number of mistakes SOA makes as M and define the rounds

[6]The term "experts" here is identical to saying hypotheses, or a subset of a hypothesis class.
[7]The inequality displayed holds whenever $T \geq \text{lit}(H) + 2$.

these mistakes were made as $1 \leq i_1, \ldots, i_M \leq T$. Observe $M \leq \text{lit}(H)$ by SOA guarantee. By construction, the predictions of $\text{Expert}(i_1, \ldots, i_M)$ differ from SOA only on rounds[8] $1 \leq i_1, \ldots, i_M \leq T$, the exact rounds on which SOA will predict the wrong label (i.e. predict $!h(x_{i_j})$ for $1 \leq j \leq M$). So on all rounds, $\text{Expert}(i_1, \ldots, i_M)$ always predicts the label $h(x_t)$ on all rounds $1 \leq t \leq T$. $\square$

The main idea here is that the finite sets of experts we have defined will essentially recover the function class. Thus, $\forall h \in H$, following our theorem there is some $i_1, \ldots, i_M$ and M such that $\text{Expert}(i_1, \ldots, i_M)$ and h have the same loss on $\{(x_i, y_i)\}_{i=1}^T$. So for any sequence $\{(x_i, y_i)\}_{i=1}^T$, we have:

$$\min_{(i_1,\ldots,i_M)} \sum_{t=1}^{T} 1[\{\text{Expert}(i_1, \ldots, i_m)(x_t) \neq y_t\}] \leq \min_{h \in H} \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\}$$
$$\implies \sum_{t=1}^{T} \mathbb{E}[1\{\hat{y}_t - y_t\}] - \min_{h \in H} \sum_{t=1}^{T} 1\{h(x_t) \neq y_t\} \leq \sqrt{(T/2) \ln N}$$

So our demonstration of Theorem 6.17 is complete. Omar notes that we can actually tightly lower bound this expected regret $\text{Reg}_H(T)$ by $O(\sqrt{\text{lit}(H) \cdot T})$:

Theorem 6.19 (Littlestone Dimension Tightly Bounds Regret)**.** *For any class H, the regret is tightly-bounded by the Littletone dimension of H,*

$$O(\sqrt{lit(H)T}) \leq Reg_H(T) \leq \tilde{O}(\sqrt{lit(H)T})$$

where $\tilde{O}(g(n))$ ignores logarithmic factors of $\log^k[g(n)]$.

So Littlestone Dimension completely characterizes hypothesis classes that are online learnable in both the realizable setting (last lecture) and agnostic setting (this section).

6.4 Connecting Boosting and Online Learning (Lecture 10)

We finish this lecture by investigating the connection between boosting and online learning. Specifically, we can derive a boosting algorithm for weak-learners using the Multiplicative Weights procedure from online learning! But we will do this in a weird way through the *dual* space: datapoints $x \in X$ are seen as the "experts" and predictors $h \in H$ will be chosen by the adversary. We formally set this up:

- There is an unknown concept $c : X \to \{\pm 1\}$

[8] Because the update step of SOA and $\text{Expert}(i_1, \ldots, i_M)$ look different, this might not be intuitive. However, their update steps are actually identical, which we now prove. Assume $V_t^{\text{SOA}} = V_t^{\text{Expert}}$ for a time step t (we are using induction). Now if $t \notin \{i_j\} \implies$ SOA correctly outputted $h(x_t)$, so the $\hat{y}_t = \tilde{y}_t = h(x_t)$ used to update V_t^{Expert} exactly matches the (correct) label $h(x_t)$ used to update V_t^{SOA}. Now if $t \in \{i_j\}$, the Expert's $\hat{y}_t = 1 - \tilde{y}_t =$ the opposite of SOA's wrong output, or just $h(x_t)$. Thus, V_t^{Expert} will be updated on $\hat{y}_t = h(x_t)$ and V_t^{SOA} will be updated on the correct label $h(x_t)$ as usual. So because the update steps are equivalent across all time steps, by induction we have shown $\forall 1 \leq t \leq T, V_t^{\text{SOA}} = V_t^{\text{Expert}}$. So we can be confident their predictions only differ on $i_1, \ldots, i_M$ as deliberately prescribed by the $\text{Expert}(i_1, \ldots, i_M)$ algorithm.

- We have access to a weak-learning algorithm B with edge γ, s.t. $\forall D$ over $X, h = B(D)$ will have $L_D(h) \leq 1/2 - \gamma$
- Initialize a distribution P_1 to be uniform over X
- Now for rounds $t = 1, 2, \ldots, T$:
  - Call weak-learner B_t on P_t, and let $h_t := B(P_t)$, where $L_{P_T}(h_t) \leq 1/2 - \gamma$
  - Update P_t distribution over X as follows:

$$P_{t+1}(x) = \frac{P_t(x) \exp\big(-\eta \mathbf{1}\{h_t(x) = c(x)\}\big)}{Z_t},$$

    where $Z_t = \sum_{x \in X} P_t(x) \exp\big(-\eta \mathbf{1}\{h_t(x) = c(x)\}\big)$ is a normalization factor.

We now demonstrate that this procedure gives us a "boosted" learner that perfectly learns $c(x)$.

Proof. We go through this step by step. The proof is not that bad.

- Let's write down the regret guarantee of Multiplicative Weights using $\eta = \sqrt{8 \ln |X|/T}$ from Corollary 6.16:

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{x \sim P_t}\big[\mathbf{1}\{h_t(x) = c(x)\}\big] - \min_{x \in X} \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}\{h_t(x) = c(x)\} \leq \sqrt{\frac{\ln |X|}{2T}}\,.$$

- Observe that $\mathbb{E}_{x \sim P_t}\big[\mathbf{1}\{h_t(x) = c(x)\}\big] = 1 - L_{P_t}(h_t) \geq \frac{1}{2} + \gamma$.
- By rearranging terms,

$$\min_{x \in X} \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}\{h_t(x) = c(x)\} \geq \tfrac{1}{2} + \gamma - \sqrt{\frac{\ln |X|}{2T}}\,.$$

- Let $T = 2 \ln |X|/\gamma^2$ so that $\sqrt{\ln |X|/(2T)} = \gamma/2$.
- This ensures that

$$\min_{x \in X} \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}\{h_t(x) = c(x)\} \geq \tfrac{1}{2} + \tfrac{\gamma}{2},$$

  so $\forall x \in X$, strictly more than 50% of the classifiers in $\{h_1, \ldots, h_t\}$ agree with $c(x)$. Thus, this implies that $\forall x \in X : \mathrm{MAJ}(h_1, \ldots, h_T)(x) = c(x)$, where MAJ is a majority vote.

□