

Lectures on Learning Theory

Lecturer: Ronitt Rubinfeld

Scribe: Yuchong Pan

1 PAC Learning

The model of *learning from random, uniform examples* is as follows: Given the *example oracle* $\text{Ex}(f)$ of a function f , pick m i.i.d. random variables x_1, \dots, x_m uniformly (or from some distribution \mathcal{D} , which might not be known to the learner in general), and call $\text{Ex}(f)$ to obtain m random labeled examples $(x_1, f(x_1)), \dots, (x_m, f(x_m))$; after seeing these examples, the learner outputs a hypothesis h of the function f .

Should we require $h = f$? This is probably too much to ask. However, we can at least require $\text{dist}(h, f) := \mathbb{P}_{x \sim \mathcal{D}}(h(x) \neq f(x)) \leq \varepsilon$, where $\text{dist}(h, f)$ is also called $\text{error}_{\mathcal{D}}(h)$ with respect to f .

Definition 1. A *uniform distribution learning algorithm* for a concept class \mathcal{C} is an algorithm \mathcal{A} that, given $\varepsilon > 0$, $\delta > 0$ and access to $\text{Ex}(f)$ for $f \in \mathcal{C}$, outputs a function h such that with probability at least $1 - \delta$, $\text{error}(h)$ with respect to f is at most ε . This is called *probably approximately correct (PAC) learning*.

We are interested in the following parameters:

- m , the *sample complexity*;
- ε , the *accuracy* parameter;
- δ , the *confidence* parameter;
- the running time, which we hope to be $\text{poly}(\log(\text{domain size}), 1/\varepsilon, 1/\delta)$;
- the *description* of h , which at least should be compact (i.e., $O(\log |\mathcal{C}|)$) and efficient to evaluate; it requires $h \in \mathcal{C}$, then this is called *proper learning*.

Note that the uniform case is a special case of the PAC model. The more general PAC model is given $\text{Ex}_{\mathcal{D}}(f)$ and bounds $\text{error}_{\mathcal{D}}(h)$ with respect to f .

2 Learning Conjunctions

Let \mathcal{C} be the class of conjunctions (i.e., 1-term DNF) over $\{0, 1\}^n$. We cannot hope for 0-error from a sub-exponential number of random samples; to see this, note that it is hard to distinguish $f(x) = x_1 \cdots x_n$ and $f(x) = \mathbf{F}$. Algorithm 1 gives a polynomial time sampling algorithm for conjunction learning, where “?” indicates a parameter to be determined.

For x_i in the conjunction, it must be set in the same way in each positive example, so $i \in V$. For x_i not in the conjunction,

$$\mathbb{P}[i \in V] = \mathbb{P}[x_i \text{ is set the same way in each of the } k \text{ positive examples}] = \frac{1}{2^{k-1}}.$$

By the union bound,

$$\mathbb{P}[\text{any } x_i \text{ not in the conjunction survives}] \leq \frac{n}{2^{k-1}} \leq \delta,$$

```

1 draw  $\text{poly}(1/\varepsilon)$  samples
2 estimate  $\mathbb{P}[f(x) = 1]$  to additive error at most  $\pm\varepsilon/4$  and confidence at least  $1 - \delta/2$ 
3 if estimate is less than  $\varepsilon/2$  then
4   return  $h(x) = 0$ 
5 (estimate is at least  $\varepsilon/2$ ; see a new positive example every  $O(1/\varepsilon)$  samples)
6 collect  $\frac{1}{\varepsilon}$  more positive examples
7  $V \leftarrow$  set of indices of variables that are set in the same way in each positive example
8 return  $h(x) = \bigwedge_{i \in V} x_i^{b_i}$ , where each  $b_i$  indicates if  $x_i$  is complemented or not

```

Algorithm 1: A polynomial time sampling algorithm for conjunction learning.

if we pick $k = \log(n/\delta)$. Therefore, if we need $\Omega(\log(n/\delta))$ positive examples, or $\Omega((1/\varepsilon) \log(n/\delta))$ total examples to rule out every x_i not in the conjunction.

3 Occam's Razor

In a high level, *Occam's Razor* claims the following:

- If we ignore the running time, then learning is easy (with a polynomial number of samples).
- The shortest explanation is the best.

To see the first claim, we consider the brute-force algorithm in Algorithm 2.

```

1 draw  $M = (1/\varepsilon)(\ln |\mathcal{C}| + \ln |1/\delta|)$ 
2 search over all  $h \in \mathcal{C}$  until find one consistent with the samples
3 return  $h$ 

```

Algorithm 2: A brute-force learning algorithm that demonstrates Occam's Razor.

We say that a function h is *bad* if $\text{error}(h)$ with respect to f is at least ε . For a bad function h ,

$$\mathbb{P}[h \text{ is consistent with the samples}] \leq (1 - \varepsilon)^M.$$

By the union bound,

$$\mathbb{P}[\text{any bad function } h \text{ is consistent with the samples}] \leq |\mathcal{C}|(1 - \varepsilon)^M = |\mathcal{C}|(1 - \varepsilon)^{\frac{1}{\varepsilon}(\ln |\mathcal{C}| + \ln |1/\delta|)} = \delta.$$

Hence, it is unlikely to output a bad hypothesis h . For example, for conjunction learning, this analysis requires $O((1/\varepsilon)(n + 1/\delta))$ samples, where Algorithm 1 has a better sample complexity. On the other hand, if we have a *good* hypothesis h ,

- (i) we can *predict* values of f on new random inputs according to distribution \mathcal{D} , since

$$\mathbb{P}_{x \sim \mathcal{D}}[f(x) = h(x)] \geq 1 - \delta;$$

- (ii) we can *compress* the description of samples $(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_m, f(x_m))$ from the naïve description which takes $m(\log |D| + \log |R|)$ bits, where D and R are the domain and the range of f , respectively, to the form “ x_1, \dots, x_m plus the description of h ” which requires $m \log |D| + \log |\mathcal{C}|$ bits only.

4 Learning via Fourier Representations

In this section, we study learning algorithms that are based on estimating the Fourier representation of a function f .

4.1 Approximating One Fourier Coefficient

Lemma 2. *For any $S \subset [n]$, one can approximate $\hat{f}(S)$ to within additive error γ (i.e., $|\text{output} - \hat{f}(S)| \leq \gamma$) with probability at least $1 - \delta$ in $O(1/\gamma^2 \log 1/\delta)$ samples.*

Proof. Recall that $\hat{f}(S) = 2\mathbb{P}_{x \in \{0,1\}^n}[f(x) \neq \chi_S(x)] - 1$. Hence, we can approximate $\hat{f}(S)$ by estimating $\mathbb{P}_{x \in \{0,1\}^n}[f(x) \neq \chi_S(x)]$ and applying the Chernoff bound. \square

4.2 Fourier Concentration and the Low Degree Algorithm

Definition 3. For all $\varepsilon \in (0, 1)$ and $n \in \mathbb{N}$, we say that a function $f : \{\pm 1\}^n \rightarrow \mathbb{R}$ has $\alpha(\varepsilon, n)$ -Fourier concentration (f.c.) if

$$\sum_{\substack{S \subset [n] \\ |S| > \alpha(\varepsilon, n)}} \hat{f}(S)^2 \leq \varepsilon.$$

For a Boolean function $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$, Parseval's identity gives $\sum_{S \subset [n]} \hat{f}(S)^2 = 1$, so having $\alpha(\varepsilon, n)$ -Fourier concentration implies that

$$\sum_{\substack{S \subset [n] \\ |S| \leq \alpha(\varepsilon, n)}} \hat{f}(S)^2 \geq 1 - \varepsilon.$$

The *low degree algorithm*, given in Algorithm 3, approximates a Boolean function with d -Fourier concentration, where $d = \alpha(\varepsilon, n)$.

```

1 take  $m = O((n^d/\tau) \log(n^d/\delta))$  samples
2 foreach  $S \subset [n]$  such that  $|S| \leq d$  do
3    $C_S \leftarrow$  estimate of  $\hat{f}(S)$ 
4 let  $h : \{\pm 1\}^n \rightarrow \mathbb{R}$  be defined by  $h(x) = \sum_{S \subset [n]: |S| \leq d} C_S \chi_S(x)$ 
5 return  $\text{sign} \circ h$  as hypothesis

```

Algorithm 3: The low degree algorithm given degree d , accuracy τ and confidence δ .

Proposition 4. *If f has d -Fourier concentration with $d = \alpha(\varepsilon, n)$, then $\mathbb{E}_{x \in \{0,1\}^n}[(f(x) - h(x))^2] \leq \varepsilon + \tau$ with probability at least $1 - \delta$.*

Proof. First, we claim that each low degree Fourier Coefficient is well approximated, i.e., with probability at least $1 - \delta$, we have $|C_S - \hat{f}(S)| \leq \gamma$ for all $S \subset [n]$ with $|S| \leq d$, where $\gamma = \sqrt{\tau/n^d}$. This can be proved using the Chernoff bound and the union bound.

Second, we show that if all low degree Fourier coefficients are well approximated, then h has a low ℓ_2 -error. Suppose $|C_S - \hat{f}(S)| \leq \gamma$ for all $S \subset [n]$ such that $|S| \leq d$. Let $g : \{\pm 1\}^n \rightarrow \mathbb{R}$ be defined by

$$g(x) = f(x) - h(x).$$

By the linearity of the Fourier transform, for all $S \subset [n]$,

$$\hat{g}(S) = \hat{f}(S) - \hat{h}(S).$$

For all $S \subset [n]$ with $|S| > d$, we have $\hat{h}(S) = 0$, so $\hat{g}(S) = \hat{f}(S)$. For all $S \subset [n]$ with $|S| \leq d$, we have $|\hat{g}(S)| \leq \gamma$, so $\hat{g}(S)^2 \leq \gamma^2$. Therefore,

$$\begin{aligned} \mathbb{E}_{x \in \{\pm 1\}^n} [(f(x) - h(x))^2] &= \mathbb{E} [g(x)^2] = \sum_{S \subset [n]} \hat{g}(S)^2 && \text{(Parseval's identity)} \\ &= \sum_{\substack{S \subset [n] \\ |S| \leq d}} \hat{g}(S)^2 + \sum_{\substack{S \subset [n] \\ |S| > d}} \hat{g}(S)^2 \\ &\leq \sum_{\substack{S \subset [n] \\ |S| \leq d}} \gamma^2 + \sum_{\substack{S \subset [n] \\ |S| > d}} \hat{f}(S)^2 \\ &\leq \binom{n}{d} \cdot \gamma^2 + \varepsilon && \text{(by Fourier concentration)} \\ &\leq \tau + \varepsilon. \end{aligned}$$

This completes the proof. \square

Proposition 5. *Let $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$. Let $h : \{\pm 1\}^n \rightarrow \mathbb{R}$. Then*

$$\mathbb{P}_{x \in \{\pm 1\}^n} [f(x) \neq \text{sign}(h(x))] \leq \mathbb{E}_{x \in \{\pm 1\}^n} [(f(x) - h(x))^2]$$

Proof. Recall that

$$\mathbb{E}_{x \in \{\pm 1\}^n} [(f(x) - h(x))^2] = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} (f(x) - h(x))^2, \quad (1)$$

$$\mathbb{P}_{x \in \{\pm 1\}^n} [f(x) \neq \text{sign}(h(x))] = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} \mathbb{1}_{f(x) \neq \text{sign}(h(x))}. \quad (2)$$

We compare (1) and (2) term by term. Let $x \in \{\pm 1\}^n$. If $f(x) = \text{sign}(h(x))$, then $\mathbb{1}_{f(x) \neq \text{sign}(h(x))} = 0 \leq (f(x) - h(x))^2$. If $f(x) \neq \text{sign}(h(x))$, then $\mathbb{1}_{f(x) \neq \text{sign}(h(x))} = 1 \leq (f(x) - h(x))^2$; see Figure 1 for an illustration.

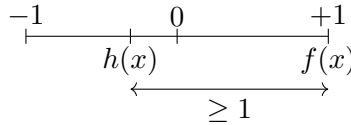


Figure 1: Illustrating the proof of Proposition 5. \square

Theorem 6. *If a concept class \mathcal{C} has Fourier concentration $d = \alpha(\varepsilon, n)$, then there exists a uniform distribution learning algorithm for \mathcal{C} with $q = O((n^d/\varepsilon) \log(n^d/\delta))$ samples; i.e., this algorithm gets q samples and with probability at least $1 - \delta$ outputs a hypothesis h' such that*

$$\mathbb{P}_{x \in \{\pm 1\}^n} [f(x) \neq h'(x)] \leq 2\varepsilon.$$

Proof. Run the low degree algorithm with $\tau = \varepsilon$. By Proposition 4, we get a hypothesis h such that $\mathbb{E}_{x \in \{\pm 1\}^n} [(f(x) - h(x))^2] \leq \varepsilon + \varepsilon = 2\varepsilon$. Let $h' = \text{sign} \circ h$. By Proposition 5, h' has error at most 2ε with respect to f . This completes the proof. \square

Following are examples of functions that have $\alpha(\varepsilon, n)$ -Fourier concentration.

- (i) Any function $f : \{\pm 1\}^n \rightarrow \mathbb{R}$ that depends on at most k variables has

$$\sum_{\substack{S \subset [n] \\ |S| > k}} \hat{f}(S)^2 = 0.$$

- (ii) Let $T = \{i_1, \dots, i_k\} \subset [n]$ be such that $|T| = k$. Let $\text{AND} : \{\pm 1\}^n \rightarrow \{\pm 1\}$ be defined by

$$\text{AND}(x) = \begin{cases} -1, & \text{if } x_i = -1 \text{ for all } i \in T, \\ 1, & \text{otherwise.} \end{cases}$$

We claim that AND has $\log(4/\varepsilon)$ -Fourier concentration. Note $\widehat{\text{AND}}(S) = 0$ for all $S \subset [n]$ with $|S| > |T|$. If $|T| \leq \log(4/\varepsilon)$, then we are done by definition. If $|T| > \log(4/\varepsilon)$, then

$$\widehat{\text{AND}}(\emptyset)^2 = (1 - 2\mathbb{P}[f(x) \neq \chi_{\emptyset}(x)])^2 = \left(1 - 2 \cdot \frac{1}{2^{|T|}}\right)^2 > 1 - \varepsilon.$$

Therefore, AND has 0-Fourier concentration.

- (iii) Let $T = \{i_1, \dots, i_k\} \subset [n]$ be such that $|T| = k$. Let $\overline{\text{AND}} : \{\pm 1\}^n \rightarrow \{\pm 1\}$ be defined by

$$\overline{\text{AND}}(x) = \begin{cases} 1, & \text{if } x_i = -1 \text{ for all } i \in T, \\ -1, & \text{otherwise.} \end{cases}$$

Let $f : \{\pm 1\}^n \rightarrow \{0, 1\}$ be defined by

$$\begin{aligned} f(x) &= \begin{cases} 1, & \text{if } x_i = -1 \text{ for all } i \in T, \\ 0, & \text{otherwise,} \end{cases} \\ &= \frac{1 - x_{i_1}}{2} \cdot \frac{1 - x_{i_2}}{2} \cdots \frac{1 - x_{i_k}}{2} \\ &= \frac{1}{2^k} \sum_{S \subset T} (-1)^{|S|} \chi_S(x). \end{aligned}$$

Note that all Fourier coefficients $\hat{f}(S)$ for $S \not\subset T$ are 0. Then

$$\overline{\text{AND}}(x) = 2f(x) - 1 = -1 + \frac{2}{2^k} + \sum_{\substack{S \subset T \\ S \neq \emptyset}} \frac{(-1)^{|S|}}{2^k} \chi_S(x).$$

- (iv) **Decision trees.** Consider a decision tree T , e.g., Figure 2. For each leaf ℓ of T , let V_ℓ denote the set of indices of variables visited on the path from the root to leaf ℓ , and let $f_\ell : \{\pm 1\}^n \rightarrow \{0, 1\}$ be defined by

$$f_\ell(x) = \prod_{i \in V_\ell} \frac{1 \pm x_i}{2} \quad (\text{"}\pm\text{" denotes a left turn or a right turn})$$

$$= \frac{1}{2^{|V_\ell|}} \sum_{S \subset V_\ell} (-1)^{\# \text{ left turns taken in } S} \chi_S.$$

Let $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ be defined by

$$f(x) = \sum_{\ell: \text{ leaf of } T} f_\ell(x) \text{val}(\ell).$$

Note that for each $x \in \{\pm 1\}^n$, exactly one of the values $f_\ell(x)$ is 1 for leaves ℓ of T , and all others are 0. Moreover, for each leaf ℓ of T , the number of variables on which f_ℓ depends is at most the depth of ℓ . By the linearity of the Fourier transform,

$$\hat{f}(S) = \sum_{\ell: \text{ leaf of } T} \hat{f}_\ell(S) \text{val}(\ell).$$

Therefore, $\hat{f}(S) = 0$ for all $S \subset [n]$ such that $|S|$ is greater than the depth of T .

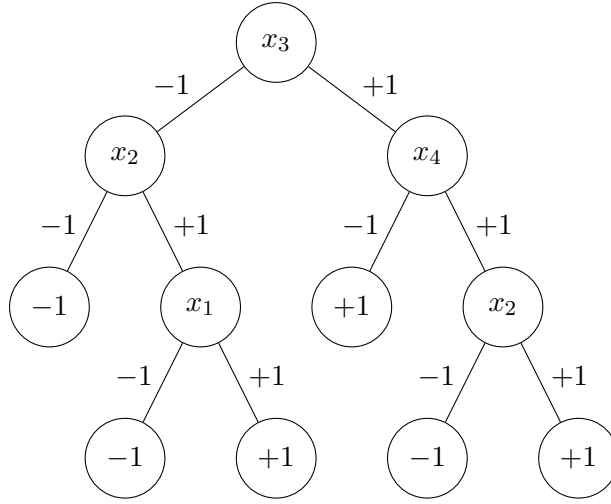


Figure 2: A decision tree.

- (v) **Constant depth circuits.** Recall that a Boolean circuit is a DAG whose vertices are gates which represent operations (e.g., \wedge, \vee, \neg), constants $(1, 0)$ and variables (x_1, \dots, x_n) . We allow each operation to have an unbounded number of inputs; the \neg gate can have only one input. Can we compute the parity (XOR) of n bits in a constant depth with a polynomial-size input for each operation? The answer is “no,” which follows from the switching lemma by Furst, Saxe and Sipser.

Theorem 7 (Hastad, Linial, Mansour and Nisan). *For any function f computable via a size- s depth- d circuit,*

$$\sum_{\substack{S \subset [n] \\ |S| > t}} \hat{f}(S)^2 \leq \alpha,$$

where $t = O(\log(s/\alpha))^{d-1}$.

Taking $s = \text{poly}(n)$, $d = O(1)$ and $\alpha = O(\varepsilon)$ implies $t = O(\log^d(n/\varepsilon))$. Therefore, the low degree algorithm gives an $n^{O(\log^d(n/\varepsilon))}$ sample algorithm for learning.

(vi) **Learning half-spaces.**

Definition 8. For $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, the function $h : \{\pm 1\}^n \rightarrow \{\pm 1\}$ defined by $h(x) = \text{sign}(w \cdot x - \theta)$ is called a *half-space function*.

Theorem 9. A half-space function $h : \{\pm 1\}^n \rightarrow \{\pm 1\}$ has Fourier concentration $\alpha(\varepsilon) = c/\varepsilon^2$ for some constant c .

Corollary 10. The low degree algorithm learns half-spaces with $n^{O(1/\varepsilon^2)}$ samples.

4.3 Noise Sensitivity

Definition 11. For $\varepsilon \in (0, 1/2)$, the *noise operator* $N_\varepsilon(x)$ randomly flips each bit of x with probability ε , given $x \in \{\pm 1\}^n$.

Definition 12. The *noise sensitivity* of a Boolean function $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ is defined to be

$$\text{NS}_\varepsilon(f) = \mathbb{P}_{\substack{x \in \{\pm 1\}^n \\ \text{noise}}} [f(x) \neq f(N_\varepsilon(x))].$$

Following are examples of the noise sensitivity of a Boolean function.

- (i) If $f(x) = x_1$, then $\text{NS}_\varepsilon(f) = \varepsilon$.
- (ii) If $f(x) = x_1 \cdots x_k$, then

$$\begin{aligned} \text{NS}_\varepsilon(f) &= \mathbb{P}[f(x) = \text{F}, f(N_\varepsilon(x)) = \text{T}] + \mathbb{P}[f(x) = \text{T}, f(N_\varepsilon(x)) = \text{F}] \\ &= 2 \mathbb{P}[f(x) = \text{T}, f(N_\varepsilon(x)) = \text{F}] \\ &= 2 \cdot \frac{1}{2^k} \cdot (1 - (1 - \varepsilon)^k). \end{aligned}$$

Therefore, if $\varepsilon \ll 1/k$, then $\text{NS}_\varepsilon(f) \approx \varepsilon k / 2^{k-1}$; if $\varepsilon \gg 1/k$, then $\text{NS}_\varepsilon(f) \approx (1 - e^{-\varepsilon k}) / 2^{k-1}$.

- (iii) If $f(x) = \text{Maj}(x_1, \dots, x_n)$, then $\text{NS}_\varepsilon(f) = O(\sqrt{\varepsilon})$.

To see this, note that $\text{Maj}(x)$ corresponds to a random walk on a line starting at 0, and that the location corresponds to the sum of the x_i 's so far.

Fact 13. If $X_1, \dots, X_n \in \{\pm 1\}$ are i.i.d random variables, then $\mathbb{E}[|X_1 + \dots + X_n|] = \sqrt{n}$, and (informally) $|X_1 + \dots + X_n|$ is likely to be close to \sqrt{n} .

Therefore, $N_\varepsilon(x)$ corresponds to a random walk on εn bits, where each flip displaces by ± 2 . By Fact 13, the expected displacement is $2\sqrt{\varepsilon n}$.

Given $x \in \{\pm 1\}^n$, we consider the following process:

1. Talk a walk specified by x .
2. Continue the walk according to $2N_\varepsilon(x)$.

Pretend that the first walk leaves us at \sqrt{n} . By Markov's inequality,

$$\begin{aligned} \mathbb{P}[\text{the second walk takes us across 0}] &= \frac{1}{2} \mathbb{P}[\text{the second displacement is greater than } \sqrt{n}] \\ &= \frac{1}{2} \cdot \frac{\mathbb{E}[\text{the second displacement}]}{\sqrt{n}} \\ &= \frac{1}{2} \cdot \frac{2\sqrt{\varepsilon n}}{\sqrt{n}} = \sqrt{\varepsilon}. \end{aligned}$$

(iv) **Linear threshold functions (half-spaces).**

Theorem 14. *If f is a linear threshold function (i.e., a half-space), then $\text{NS}_\varepsilon(f) < 8.8\sqrt{\varepsilon}$.*

(v) **Parity functions.**

Proposition 15. *Let $S \subset [n]$ be such that $|S| = k$. Then*

$$\text{NS}_\varepsilon(\chi_S) = \frac{1 - (1 - 2\varepsilon)^k}{2}.$$

(vi) **Any function.**

Theorem 16. *For any $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$,*

$$\text{NS}_\varepsilon(f) = \frac{1}{2} - \frac{1}{2} \sum_{S \subset [n]} (1 - 2\varepsilon)^{|S|} \hat{f}(S)^2.$$

The proof of Theorem 16 is in homework.

Next, we show the relation between noise sensitivity and Fourier concentration.

Theorem 17. *For any $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ and $\gamma \in (0, 1/2)$,*

$$\sum_{\substack{S \subset [n] \\ |S| \geq \frac{1}{\gamma}}} \hat{f}(S)^2 < 2.32 \text{NS}_\gamma(f).$$

Proof. We have

$$\begin{aligned} 2 \text{NS}_\gamma(f) &= 1 - \sum_{S \subset [n]} (1 - 2\gamma)^{|S|} \hat{f}(S)^2 && \text{(Theorem 16)} \\ &= \sum_{S \subset [n]} \left(\hat{f}(S)^2 - (1 - 2\gamma)^{|S|} \hat{f}(S)^2 \right) && \text{(Boolean Parseval's identity)} \\ &= \sum_{S \subset [n]} \left(1 - (1 - 2\gamma)^{|S|} \right) \hat{f}(S)^2 \\ &\geq \sum_{\substack{S \subset [n] \\ |S| \geq \frac{1}{\gamma}}} \left(1 - (1 - 2\gamma)^{\frac{1}{\gamma}} \right) \hat{f}(S)^2 \\ &> \sum_{\substack{S \subset [n] \\ |S| \geq \frac{1}{\gamma}}} (1 - e^{-2}) \hat{f}(S)^2. \end{aligned}$$

Therefore,

$$\sum_{\substack{S \subset [n] \\ |S| \geq \frac{1}{\gamma}}} \hat{f}(S)^2 \leq \text{NS}_\gamma(f) \left(\frac{1}{1 - e^{-2}} \right) < 2.32 \text{NS}_\gamma(f).$$

This completes the proof. □

Corollary 18. For any half-space $h : \{\pm 1\}^n \rightarrow \{\pm 1\}$,

$$\sum_{\substack{S \subset [n] \\ |S| \geq O(1/\varepsilon^2)}} \hat{f}(S)^2 \leq \varepsilon.$$

Therefore, one can learn any half-space from $n^{O(1/\varepsilon^2)}$ random examples.

Corollary 19. Any function of k half-spaces can be learned with $n^{O(k^2/\varepsilon^2)}$ samples.

4.4 Learning Heavy Fourier Coefficients

Given $\theta > 0$ and black box access to $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$, we want to

- (i) output all coefficients $S \subset [n]$ such that $|\hat{f}(S)| \geq \theta$;
- (ii) only output coefficients $S \subset [n]$ such that $|\hat{f}(S)| \geq \theta/2$.

The main idea is *exhaustive search with good pruning*. The search tree consists of n levels, each representing a variable x_i for $i \in [n]$. For each level $i \in [n]$, going left means $x_i \in S$, and going right means $x_i \notin S$. Informally, we only go down paths with lots of “weights,” and we output leaves reached at the bottom level.

Fix $k \in \{0, \dots, n\}$ representing the current level of search. Fix $S_1 \subset [k]$ representing the current node of search. Let $f_{k,S_1} : \{\pm 1\}^{n-k} \rightarrow \mathbb{R}$ be defined by

$$f_{k,S_1}(x) = \sum_{T \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2) \chi_{T_2}(x).$$

Note that we could replace χ_{T_2} with $\chi_{S_1 \cup T_2} = \chi_{S_1} \cdot \chi_{T_2}$, but χ_{S_1} remains the same. If $k = 0$, then

$$f_{0,\emptyset}(x) = \sum_{T_2 \subset [n]} \hat{f}(T_2) \chi_{T_2}(x) = f(x).$$

On the other hand, if $k = n$, then for any $S_1 \subset [n]$,

$$f(n, S_1)(x) = \sum_{T_2 \subset \emptyset} \hat{f}(S_1 \cup T_2) \chi_{T_2}(x) = \hat{f}(S_1).$$

The plan is to only go down paths with $\mathbb{E}[f_{k,S_1}(x)^2] \geq \theta^2$. There are several questions to answer:

- (i) Can we compute it?
- (ii) Does it bring us to right leaves? In other words, do we get to all heavy leaves, and do we get to junks (i.e., light leaves)?
- (iii) How many paths do we take? In other words, are there a lot of dead ends, and is the running time good?

First, we show that there are not too many paths.

Lemma 20. Let $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$.

- (i) At most $1/\theta^2$ sets $S \subset [n]$ satisfy $|\hat{f}(S)| \geq \theta$.
- (ii) For all $k \in \{0, \dots, n\}$, at most $1/\theta^2$ functions f_{k,S_1} have $\mathbb{E}_{x \in \{\pm 1\}^{n-k}}[f_{k,S_1}(x)^2] \geq \theta^2$.

Lemma 20 implies that our algorithm explores at most $O(1/\theta^2)$ nodes of the search tree.

Proposition 21. *For any $k \in \{0, \dots, n\}$ and $S_1 \subset [n]$,*

$$\mathbb{E}_{x \in \{\pm 1\}^{n-k}} [f_{k,S_1}(x)^2] = \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2)^2.$$

Proof. we have

$$\begin{aligned} \mathbb{E}_{x \in \{\pm 1\}^{n-k}} [f_{k,S_1}(x)^2] &= \mathbb{E}_{x \in \{\pm 1\}^{n-k}} \left[\left(\sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2) \chi_{T_2}(x) \right)^2 \right] \\ &= \mathbb{E}_{x \in \{\pm 1\}^{n-k}} \left[\sum_{T_2, T'_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2) \hat{f}(S_1 \cup T'_2) \chi_{T_2}(x) \chi_{T'_2}(x) \right] \\ &= \sum_{T_2, T'_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2) \hat{f}(S_1 \cup T'_2) \mathbb{E}_{x \in \{\pm 1\}^{n-k}} [\chi_{T_2}(x) \chi_{T'_2}(x)]. \end{aligned}$$

Note that

$$\mathbb{E}_{x \in \{\pm 1\}^{n-k}} [\chi_{T_2}(x) \chi_{T'_2}(x)] = \begin{cases} 0, & \text{if } T_2 \neq T'_2, \\ 1, & \text{if } T_2 = T'_2. \end{cases}$$

Therefore,

$$\mathbb{E}_{x \in \{\pm 1\}^{n-k}} [f_{k,S_1}(x)^2] = \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2)^2.$$

This completes the proof. □

Proof of Lemma 20. (i) By the Boolean Parseval's identity,

$$1 = \mathbb{E}_{x \in \{\pm 1\}^n} [f(x)^2] = \sum_{S \subset [n]} \hat{f}(S)^2.$$

Therefore, if more than $1/\theta^2$ sets $S \subset [n]$ had $|\hat{f}(S)| \geq \theta$, then

$$\sum_{S \subset [n]} \hat{f}(S)^2 > \frac{1}{\theta} \cdot \theta^2 = 1,$$

a contradiction.

(ii) Let $k \in \{0, \dots, n\}$.

Therefore,

$$\begin{aligned} 1 &= \sum_{S \subset [n]} \hat{f}(S)^2 && \text{(Boolean Parseval's identity)} \\ &= \sum_{S_1 \subset [k]} \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2)^2 \\ &= \sum_{S_1 \subset [k]} \mathbb{E}_{x \in \{\pm 1\}^{n-k}} [f_{k,S_1}(x)^2] && \text{(Proposition 21)} \end{aligned}$$

Therefore, at most $1/\theta^2$ sets $S_1 \subset [k]$ have $\mathbb{E}_{x \in \{\pm 1\}^{n-k}} [f_{k,S_1}(x)^2] \geq \theta^2$. □

Second, we show that the algorithm does not miss out heavy Fourier coefficients. To see this, for any $k \in \{0, \dots, n\}$ and $S_1 \subset [k]$, if there exists $T_2^* \subset \{k+1, \dots, n\}$ such that $|\hat{f}(S_1 \cup T_2^*)| > \theta$, then Proposition 21 implies that

$$\mathbb{E}_{x \in \{\pm 1\}^n} [f_{k, S_1}(x)^2] = \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2)^2 \geq \hat{f}(S_1 \cup T_2^*)^2 \geq \theta^2.$$

To avoid outputting junks (i.e., light Fourier coefficients), we add a simple fix to the algorithm: For each candidate to S , estimate its Fourier coefficient and make sure it is big enough before outputting it.

Lemma 22. For $x \in \{\pm 1\}^{n-k}$,

$$f_{k, S_1}(x) = \mathbb{E}_{y \in \{\pm 1\}^k} [f(yx) \chi_{S_1}(y)],$$

where yx denotes the concatenation operation.

By Lemma 22, we can estimate $f_{k, S_1}(x)$ by picking random y , outputting the average and using the Chernoff bound.

Proof. Let $x \in \{\pm 1\}^{n-k}$. Then

$$f(yx) = \sum_{T \subset [n]} \hat{f}(T) \chi_T(yx).$$

Since $T = T_1 \cup T_2$ where $T_1 \subset [k]$ and $T_2 \subset \{k+1, \dots, n\}$, then $\chi_T(yx) = \chi_{T_1}(y) \chi_{T_2}(x)$. Therefore,

$$\begin{aligned} \mathbb{E}_{y \in \{\pm 1\}^k} [f(yx) \chi_{S_1}(y)] &= \mathbb{E}_{y \in \{\pm 1\}^k} \left[\sum_{T_1 \subset [k]} \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(T_1 \cup T_2) \chi_{T_1}(y) \chi_{T_2}(x) \chi_{S_1}(y) \right] \\ &= \sum_{T_1 \subset [k]} \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(T_1 \cup T_2) \chi_{T_2}(x) \mathbb{E}_{y \in \{\pm 1\}^k} [\chi_{T_1}(y) \chi_{S_1}(y)]. \end{aligned}$$

Note that

$$\mathbb{E}_{y \in \{\pm 1\}^k} [\chi_{T_1}(y) \chi_{S_1}(y)] = \begin{cases} 0, & \text{if } T_1 \neq S_1, \\ 1, & \text{if } T_1 = S_1. \end{cases}$$

Therefore,

$$\mathbb{E}_{y \in \{\pm 1\}^k} [f(yx) \chi_{S_1}(y)] = \sum_{T_2 \subset \{k+1, \dots, n\}} \hat{f}(S_1 \cup T_2) \chi_{T_2}(x) = f_{k, S_1}(x).$$

This completes the proof. \square

Now we are ready to summarize the full algorithm, called the Kushilevitz-Mansour algorithm, in Algorithm 4.

The Kushilevitz-Mansour algorithm has the following applications:

- (i) Learn decision trees of size $\leq t$ (not just small depth);
- (ii) functions of small L_1 -norm, where the L_1 -norm of a function $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ is defined to be $L_1(f) = \sum_{S \subset [n]} |\hat{f}(S)|$ (by setting $\theta \leftarrow \varepsilon/L_1(f)$).

```

1 if  $k = n$  then (leaf)
2   if the  $\theta/4$ -additive estimate of  $\hat{f}(S_1)$  is at least  $3/4$  then
3     return  $S_1$ 
4 else if  $\mathbb{E}_{x \in \{\pm 1\}^{n-k}}[f_{k, S_1 \cup \{k+1\}}(x)^2] \geq \theta^2/2$  then (check left subtree)
5   recurse on  $(k+1, S_1 \cup \{k+1\})$ 
6 else if  $\mathbb{E}_{x \in \{\pm 1\}^{n-k}}[f_{k, S_1}(x)^2] \geq \theta^2/2$  then (check right subtree)
7   recurse on  $(k+1, S_1)$ 

```

Algorithm 4: The Kushilevitz-Mansour algorithm, given the current level $k \in \{0, \dots, n\}$ and the current prefix $S_1 \subset [n]$.

5 Monotone Functions

Definition 23. We define \preceq to be a partial order over $\{\pm 1\}^n$ such that $x \leq y$ if and only if $x_i \leq y_i$ for all $i \in [n]$.

Definition 24. A Boolean function $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ is *monotone* if $f(x) \leq f(y)$ for all $x, y \in \{\pm 1\}^n$ with $x \preceq y$.

Are monotone functions learnable? Occam's Razor implies that $O(\log |\mathcal{C}|)$ samples suffice, where \mathcal{C} is the class of monotone functions. We have the following lower bound on $|\mathcal{C}|$: Consider *slice functions* which have values $+1$ on level greater than $n/2$ in the Boolean hypercube, values -1 on level less than $n/2$, and arbitrary values on level $n/2$. Then slice functions are all monotone. There are $\binom{n}{n/2}$ nodes on level $n/2$, so there are $2^{\binom{n}{n/2}}$ monotone functions. Therefore, the sample complexity implied by Occam's Razor is $O(\binom{n}{n/2}) = O(2^n / \sqrt{n})$. In homework, we shall show that $2^{O(\sqrt{n})}$ samples suffice.

We show that monotone functions have weak agreement with some dictator function.

Theorem 25. For any monotone Boolean function $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$, there exists $g \in S := \{\pm 1, x_1, x_2, \dots, x_n\}$ such that

$$\mathbb{P}_{x \in \{\pm 1\}^n} [f(x) = g(x)] \geq \frac{1}{2} + \Omega\left(\frac{1}{n}\right).$$

Note that if we add the majority function to S , then we can get $1/2 + O(1/\sqrt{n})$. Theorem 25 suggests the following learning algorithm: Estimate the agreement of f with each member of S , and output the best.

If f has weak agreement with $+1$ or -1 , then we are done. Otherwise, $\mathbb{P}[f(x) = 1] \in [1/4, 3/4]$. Consider a “graphic” view of the Boolean hypercube in Figure 3, where vertices with value 1 are colored red and vertices with value -1 are colored blue.

Definition 26. Let $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ be a monotone Boolean function. The *influence of f at the i^{th} direction* is defined to be

$$\text{Inf}_i(f) = \frac{\# \text{ red-blue edges in the } i^{\text{th}} \text{ direction}}{2^{n-1}} = \mathbb{P}_{x \in \{\pm 1\}^n} [f(x) \neq f(x^{\oplus i})],$$

where $x^{\oplus i}$ is the vector obtained by flipping the i^{th} bit in x . The *influence of f* is defined to be

$$\text{Inf}(f) = \frac{\# \text{ red-blue edges}}{2^n} = \sum_{i=1}^n \text{Inf}_i(f).$$

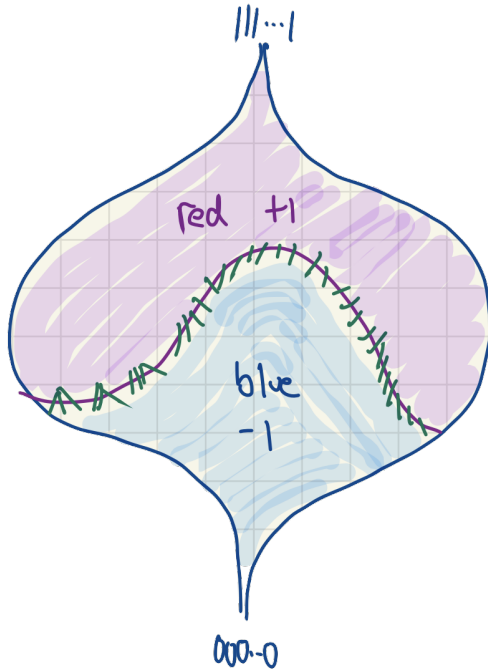


Figure 3: A “graphic” view of the Boolean hypercube, where vertices with value 1 are colored red and vertices with value -1 are colored blue.

Theorem 27. Any monotone Boolean function has $\text{Inf}_i(f) = \hat{f}(\{i\})$.

Theorem 28. The majority function $f(x) := \text{sign}(\sum_{i=1}^n x_i)$, where n is odd, maximizes the influence among monotone functions.

Theorems 27 and 28 will be proved in homework. By Theorem 27, any monotone Boolean function f has

$$\text{Inf}_i(f) = \hat{f}(\{i\}) = 2 \mathbb{P}_{x \in \{\pm 1\}^n} [f(x) = \chi_{\{i\}}(x)] - 1 = 2 \mathbb{P}_{x \in \{\pm 1\}^n} [f(x) = x_i] - 1.$$

Therefore, showing $\text{Inf}_i(f) \geq \Omega(1/n)$ is equivalent to showing

$$\mathbb{P}_{x \in \{\pm 1\}^n} \geq \frac{1}{2} + \frac{\text{Inf}_i(f)}{2} = \frac{1}{2} + \Omega\left(\frac{1}{n}\right).$$

Hence, such an index $i \in [n]$ would prove Theorem 25.

We introduce the *canonical path argument*, which proceeds as follows:

- (i) Define the canonical path for every red-blue pair of vertices (which must cross at least one red-blue edge).
- (ii) Show an upper bound on the number of canonical paths passing through any edge (called *bounding the congestion*).
- (iii) Conclude a lower bound on the number of red-blue edges.

First, we define the canonical path for every red-blue pair of vertices.

Definition 29. For all $(x, y) \in \{0, 1\}^n \times \{0, 1\}^n$ such that x is red and y is blue, the *canonical path* from x to y is defined by the following procedure: Scan bits from left to right, and flip where needed. Each flip corresponds to a step in the canonical path (in the undirected Boolean hypercube).

Note that if x is red and y is blue, then the canonical path from x to y must include some red-blue edge. Recall that $\mathbb{P}[f(x) = 1] \in [1/4, 3/4]$, then the number of red-blue (x, y) pairs that have canonical paths is at least

$$\left(\frac{1}{4} \cdot 2^n\right) \cdot \left(\frac{1}{4} \cdot 2^n\right) = \frac{1}{16} \cdot 2^{2n}.$$

Second, we show an upper bound on the number of canonical paths passing through any (red-blue) edge. Note that any red-blue pair (x, y) taking edge $(u, u^{\oplus i})$ must have

$$\begin{aligned} y_1 \dots y_{i-1} y_i &= u_1 \dots u_{i-1} \overline{u_i}, \\ x_i x_{i+1} \dots x_n &= u_i u_{i+1} \dots u_n. \end{aligned}$$

Hence, we can pick y_{i+1}, \dots, y_n and x_1, \dots, x_{i-1} freely, so there are at most 2^n total choices.

Third, since each red-blue canonical path crosses at least one red-blue edge,

$$(\# \text{ red-blue edges}) \cdot (\max \# \text{ canonical paths using each edge}) \geq \# \text{ red-blue canonical paths}.$$

Therefore, the number of red-blue edges is at least

$$\frac{\frac{1}{16} \cdot 2^{2n}}{2^n} = \frac{1}{16} \cdot 2^n.$$

It follows that there exists $i \in [n]$ such that there exist at least $2^n/(16n)$ red-blue edges in the i^{th} direction, so

$$2 \mathbb{P}_{x \in \{\pm 1\}^n} [f(x) = x_i] - 1 = \hat{f}(\{i\}) = \text{Inf}_i(f) \geq \frac{2^n}{16n} = \frac{1}{8n}.$$

This proves that there exists $i \in [n]$ such that $\mathbb{P}_{x \in \{\pm 1\}^n} [f(x) = x_i] \geq 1/2 + 1/(16n)$.

6 Weak and Strong Learning

Definition 30. An algorithm \mathcal{A} *weakly PAC learns* a concept class \mathcal{C} if there exists $\gamma > 0$ such that for any $C \in \mathcal{C}$, distribution \mathcal{D} and $\delta > 0$, with probability at least $1 - \delta$, given $\text{Ex}_{\mathcal{D}}(C)$, \mathcal{A} outputs h such that

$$\text{error}_{\mathcal{D}}(h) := \mathbb{P}_{x \sim \mathcal{D}} [h(x) \neq C(x)] \leq \frac{1}{2} - \frac{\gamma}{2}. \quad (3)$$

Here, $\gamma/2$ is called the *advantage over guessing*.

Note that we used to ask for $\mathbb{P}_{x \sim \mathcal{D}} [h(x) = C(x)] \geq 1 - \varepsilon$ for all $\varepsilon > 0$; this is called (*strong*) *PAC learning*.

Theorem 31. *If a concept class \mathcal{C} can be weakly learned on any distribution \mathcal{D} , then \mathcal{C} can be (strongly) learned, i.e., with error at most ε for all $\varepsilon > 0$.*

We are interested in the dependence of the running time of learning on γ , δ and ε . The plan for the proof of Theorem 31 is as follows:

- (i) “Modest” accuracy boosting.
- (ii) Recursively use (i) to drive down error.

First, we present a modest accuracy boosting algorithm in Algorithm 12, given an example oracle $\text{Ex}_{\mathcal{D}}(f)$ to f with distribution \mathcal{D} and a weak learner WL. Let

$$\begin{aligned}\beta_1 &= \mathbb{P}_{x \in \mathcal{D}} [h_1(x) \neq f(x)], \\ \beta_2 &= \mathbb{P}_{x \in \mathcal{D}_2} [h_2(x) \neq f(x)], \\ \beta_3 &= \mathbb{P}_{x \in \mathcal{D}_3} [h_3(x) \neq f(x)],\end{aligned}$$

We observe that if $h_1(x) = f(x)$, then $\mathcal{D}(x) = 2(1 - \beta_1)\mathcal{D}_2(x)$ (i.e., $\mathcal{D}_2(x) = \mathcal{D}(x) \cdot 1/(2(1 - \beta_1))$); if $h_1(x) \neq f(x)$, then $\mathcal{D}(x) = 2\beta_1\mathcal{D}_2(x)$. See Figure 4 for an illustration. Moreover, we divide \mathcal{D}_2 into four regions in Table 1, also illustrated in Figure 4. We have the following lemma, the proof of which we omit for now.

Lemma 32. $\text{error}_{\mathcal{D}}(h) \leq g(\beta) := 3\beta^3 - 2\beta^3$.

```

1  $h_1 \leftarrow$  run WL on  $\mathcal{D}$  for function  $f$ 
2 create a distribution  $\mathcal{D}_2$  by flipping a coin such that
3   if head then
4     draw examples from  $\mathcal{D}$  until finding  $x$  such that  $h_1(x) = f(x)$  ( $h_1$  is correct)
5     output  $x$ 
6   else
7     draw examples from  $\mathcal{D}$  until finding  $x$  such that  $h_1(x) \neq f(x)$  ( $h_1$  is incorrect)
8     output  $x$ 
9  $h_2 \leftarrow$  run WL on  $\mathcal{D}_2$  for function  $f$  ( $\text{error}_{\mathcal{D}_2}(h_2) = 1/2$ )
10 create a distribution  $\mathcal{D}_3$  by drawing examples from  $\mathcal{D}$  until finding  $x$  with  $h_1(x) \neq h_2(x)$ 
11  $h_3 \leftarrow$  run WL on  $\mathcal{D}_3$  for function  $f$ 
12 return  $\text{Maj}(h_1, h_2, h_3)$  (on  $x$ , compute  $h_1(x), h_2(x), h_3(x)$ , output majority answer)

```

Algorithm 5: A modest accuracy boosting algorithm, given an example oracle $\text{Ex}_{\mathcal{D}}(f)$ to f with distribution \mathcal{D} and a weak learner WL.

R_1	$h_1(x) = h_2(x) = f(x)$	$\text{Maj}(h_1, h_2, h_3)$ is correct
R_2	$h_1(x) = f(x) \neq h_2(x)$	maybe h_3 helps?
R_3	$h_1(x) = h_2(x) \neq f(x)$	$\text{Maj}(h_1, h_2, h_3)$ is incorrect
R_4	$h_1(x) \neq h_2(x) = f(x)$	maybe h_3 helps?

Table 1: Four regions of \mathcal{D}_2 .

Second, we give a recursive accuracy boosting algorithm in Algorithm 6. We illustrate the recursion from Algorithm 6 in Figure 5.

Corollary 33. *If a concept class \mathcal{C} is learnable, then any concept $C \in \mathcal{C}$ has a polynomial size circuit.*

Theorem 34. *Suppose that a function f cannot be computed by polynomial size circuits. Then there exists a distribution \mathcal{D} such that f is average-case hard on \mathcal{D} (i.e., there exists no polynomial size circuit that gets it right more than $1/2$ in time $1/\text{poly}(n)$).*

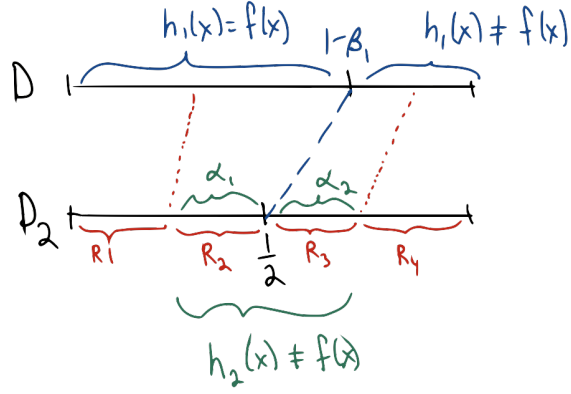


Figure 4: Illustrating the analysis of Algorithm 12.

```

1 if  $\rho$  is already good enough then
2   return WL on  $\mathcal{D}'$ 
3  $\beta \leftarrow g^{-1}(\rho)$  (error bound required from below to get error  $\rho$ )
4 define  $\mathcal{D}'_2$  and  $\mathcal{D}'_3$  as in Algorithm 12
5 foreach  $i \in [3]$  do
6    $h_i \leftarrow \text{STRONGLEARN}(\beta, \mathcal{D}'_i)$ 
7 return Maj( $h_1, h_2, h_3$ )

```

Algorithm 6: A recursive accuracy boosting algorithm STRONGLEARN, given required error ρ and distribution \mathcal{D}' .

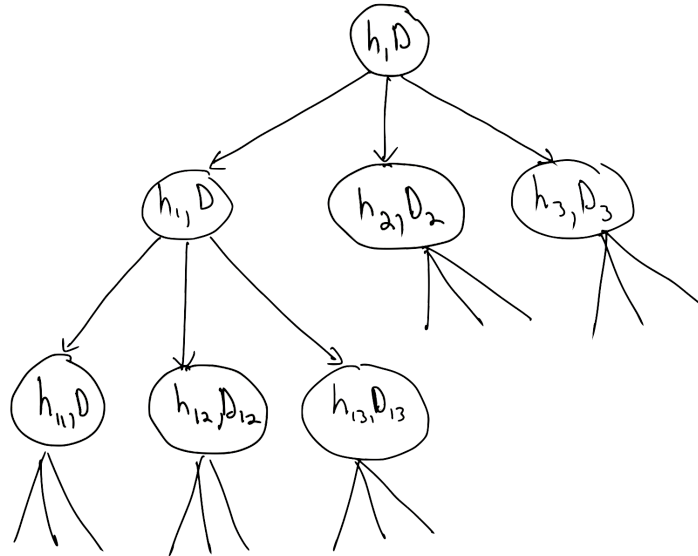


Figure 5: Illustrating the recursion of Algorithm 6.