An *and-or circuit* with n inputs is a directed acyclic graph with one sink. Each source node of the graph is labeled by one of the literals $\{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\}$, while the other nodes are labeled by AND or OR. An and-or circuit computes a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ in the natural way: The values of the inputs are substituted at the sources and propagated along the circuit until the value of the output is found at the sink.

The *size* of an and-or circuit is its number of nodes. Its *depth* is the maximum length of a directed path. A circuit of depth one is just an AND of literals or an OR of literals. Circuits of depth two are formulas in conjunctive normal form (AND of ORs of literals) or disjunctive normal form (OR of ANDs of literals). It is easy to see that any boolean function can be represented as a CNF or DNF. However, even some natural ones, like the XOR of n bits, require depth two circuits of size exponential in n.

Unlike for many other topics, complexity theory has a great deal to say about the computational power of circuits of small depth that are not too large. One very useful tool in this area is a theorem of Linial, Mansour, and Nisan which concerns the Fourier coefficients of (functions computed by) shallow circuits.

Theorem 1. *Let $f\colon \{0,1\}^n \to \{1,-1\}$ be a function computable by an and-or circuit of size s and depth d. Then for every t,*

$$\sum_{a\colon |a|>t} \hat{f}_a^2 \leq 2s \cdot 2^{-\Omega(t^{1/d})}.$$

Here $|a|$ denotes the hamming weight of a. To understand this theorem it helps to think of the fourier coefficients as partially ordered by the hamming weight of the index a; thus the "large" fourier coefficients are those $\hat{f}_a$ where $|a|$ is large. The theorem tells us that when d and s are small, f cannot have much of its fourier weight concentrated on large coefficients.

For example, take $d = 10$, $s = n^{100}$ and $t = K(\log n)^d$, where K is a constant chosen so that $2s2^{-\Omega(t^{1/d})} < 1$. Then the theorem tells us that and-or circuits of size n^{100} and depth 10 cannot compute any function all of whose nonzero fourier coefficients are over sets of size $K(\log n)^d$ or more. In particular, it says that no such circuit can compute a character, i.e. a linear function over $K(\log n)^d$ or more bits.

1 Shallow circuits can be learned

The objective of learning theory is to understand how a function f behaves in general by observing various inputs and outputs of f. There are various models depending on which input-output pairs are available (arbitrary, random), whether the information is always accurate or noisy, and what kind of answer want (reconstruct f completely, approximate f on random inputs, and so on).

The model we will be interested here is the following one. We have access to a button and every time we push it we get a "random example" of the form $(X, f(X))$. After we are finished looking at the random examples we want to output a "hypothesis" g that matches f on almost all the inputs.

How can we go about this? Without any additional information about f, it seems that we need to observe the value of f everywhere – or at least on a large part of its 2^n inputs – before we can deduce anything about how f will behave on a random input.

What if we know that f is computable by a circuit of size s and depth d? In general, if we know that f comes from some restricted family of functions $\mathcal{F}$, the following extrapolation algorithm does a lot better. After observing sufficiently many samples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$, look at all the functions g that are consistent with the information and output the one that agrees with f most often. It is not hard to show that with probability at least $2/3$, g will agree with f on a $1-\eta$ fraction of inputs as long as m is at least on the order of $(\log|\mathcal{F}|)/\eta$. If $\mathcal{F}$ is the family of circuits of size s (and depth d), then $|\mathcal{F}| = 2^{s \log n + O(s)}$, and f can be learned after seeing $m = O(s \log n/\eta)$ samples. The only issue is that looking for a function g in this gigantic family of functions takes a lot of time. Is it possible to do better?

Theorem 1 indicates that if f is computable by a sufficiently shallow and small circuit, then the large fourier coefficients of f should not matter too much. So if we could "learn" the small fourier coefficients, we might hope to have a good approximation of f. Since the fourier coefficients merely represent the correlation of f with various characters, we could hope to get fairly accurate estimates of them by sampling. Suppose you observe random samples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$. Then you can estimate the fourier coefficient $\hat{f}_a$ as the average of the values $f(x_i)\chi_a(x_i)$. If the samples are independent, the accuracy of this estimate is determined by the Chernoff bound, which says that $m = O((1/\varepsilon^2)\log(1/\delta))$ samples are sufficient to guarantee that

$$\Pr[|\alpha_a - \hat{f}_a| > \varepsilon] \leq \delta.$$

By a union bound,

$$\Pr[|\alpha_a - \hat{f}_a| \leq \varepsilon \text{ for all } a \text{ with } |a| \leq t] \geq 1 - \delta\binom{n}{t}.$$

So suppose we have estimates α_a such that $|\alpha_a - \hat{f}_a| \leq \varepsilon$ for all a of weight at most t. Since Theorem 1 tells us that the larger fourier coefficients should not matter much, it is tempting to approximate f by the function

$$g(x) = \sum_{a\colon |a| \leq t} \alpha_a \chi_a(x).$$

The only problem is that this function is not boolean – it could output arbitrary real values. Somewhat surprisingly, an easy fix to this problem works: Round g to the closest boolean function.

Lemma 2. *For every pair of functions $f\colon \{0,1\}^n \to \{1,-1\}$ and $g\colon \{0,1\}^n \to \mathbb{R}$,*

$$\Pr[f(x) \neq \operatorname{sign} g(x)] \leq \sum_{a \in \{0,1\}^n} (\hat{f}_a - \hat{g}_a)^2$$

where $\operatorname{sign} t$ *takes value* 1 *when* t *is nonnegative and value* -1 *when* t *is negative.*

Proof. If $f(x) \neq \operatorname{sign} g(x)$, then $|f(x) - g(x)| \geq 1$, so

$$\Pr[f(x) \neq \operatorname{sign} g(x)] \leq \mathrm{E}[|f(x) - g(x)|^2] = \sum_a (\hat{f}_a - \hat{g}_a)^2.$$

The second equality is Parseval's identity applied to the function $f - g$. □

It remains to choose the parameters. Assuming all the estimates of $\hat{f}_a$ are accurate within ε, we get

$$\sum_a (\hat{f}_a - \hat{g}_a)^2 \leq \binom{n}{t}\varepsilon^2 + 2s \cdot 2^{-\Omega(t^{1/d})}$$

To ensure that $\Pr[f(x) \neq \operatorname{sign} g(x)] \leq \eta$, we want to choose t and ε so that $\binom{n}{t}\varepsilon^2 \leq \eta/2$ and $2s \cdot 2^{-\Omega(t^{1/d})} \leq \eta/2$. This is achieved for $t = \log(O(s/\eta))^d$ and $\varepsilon = \sqrt{\eta/2n^t}$. Finally, if we set $\delta = 1/3n^t$, we have proved the following theorem:

Theorem 3. *Assume $f\colon \{0,1\}^n \to \{1,-1\}$ can be computed by a circuit of size s and depth d. Then with probability $2/3$ over the choice of samples $x_1, \ldots, x_m \sim \{0,1\}^n$, where $m = n^{\log(O(s/\eta))^d}$, the functions $\operatorname{sign} g(x)$ and $f(x)$ match on a $1-\eta$ fraction of inputs.*

In other words, by observing $m = n^{\log(O(s/\eta))^d}$ random samples $(X, f(X))$, we can come up with a prediction that agrees with f on $1-\eta$ fraction of inputs, with probability $2/3$ over the choice of samples. Calculating and evaluating the prediction take time $m = n^{\log(O(s/\eta))^d}$. This is a bit worse in terms of sample complexity than the extrapolation algorithm, but a lot better in terms of time complexity.

2 Shallow circuits and random restrictions

The main tool in the proof of Theorem 1 is the a fundamental result by Håstad (improving on previous works by others) about how shallow circuits react under random restrictions. Let ρ be a string in $\{0,1,\star\}^n$ and f a function on n bits. The restricted function f^ρ on f is obtained by partially assigning the inputs that are labeled by 0 or 1 in ρ by these values and leaving the outputs unassigned. A p-random restricted function is a distribution on functions f^ρ, where each bit of ρ is chosen uniformly independently at random to have value 0 or 1 with probability $(1-p)/2$ each and $\star$ with probability p.

It turns out that random restrictions (with p sufficiently small) have a simplifying effect on functions computable by shallow circuits. To illustrate this take an AND of k literals. A random restriction fixes the value of this AND with probability at least $1 - [(1-p)/2]^k$, for it is sufficient that at least one of the literals is fixed to zero by the restriction. The same argument applies to an OR of k literals. Thus if k is large, a random restriction is likely to completely kill off a circuit of depth one; the circuit can survive only if it depends on a few inputs to begin with.

This can be generalized to larger depths. To explain the generalization we need to say a bit about one other model. A *decision tree* is a binary boolean tree where every internal node is labeled by a variable x_i and its outgoing edges are labeled by 1 and -1. The leaves are labeled by $0, 1$ values. The decision tree computes a function $f\colon \{0,1\}^n \to \{1,-1\}$ by following the path from the root to the leaves labeled by the input values and outputting the value at the leaf.

Theorem 4. *Suppose $f\colon \{0,1\}^n \to \{1,-1\}$ is computable by a circuit of size s and depth d. Then for every k,*

$$\Pr[f^\rho \text{ has a decision tree of depth } k] \geq 1 - s2^{-k}$$

where the probability is taken over a choice of p-random restriction with $p = 1/(10k)^{d-1}$.

The exponential dependence of p on d (in fact the exact value of the exponent $d-1$) is tight. If a function g is computable by a decision tree of depth k, then $\hat{g}_a = 0$ for all a of weight more than k. To see this, we associate to every leaf ℓ of the decision tree that is labeled by 1 a function g^ℓ so that $g^\ell(x) = 1$ when and only when input x follows the path ℓ. Then $g^\ell(x)$ is the product of at most k terms of the form x_i of $1 - x_i$. Since g^ℓ depends on at most k terms, we must have $\hat{g}^\ell_a = \mathrm{E}[g^\ell(x)\chi_a(x)] = 0$ for all a of weight more than k. We can now write

$$g(x) = 1 - 2 \sum_{\ell \text{ labeled by } 1} g_\ell(x).$$

and by linearity we get that $\hat{g}_a = 0$ for all a of weight more than k as well.

Now Theorem 4 tells us that a randomly restricted version of f is unlikely to have large nonzero fourier coefficients. How can we translate this to a statement about f itself?

Let's first fix a subset a coordinates $V \subseteq [n]$ and an assignment $v \in \{0,1\}^V$ on those coordinates. Without loss of generality, let's assume V consists of the first i coordinates. In the last lecture we showed that

$$\sum_{a \in FIX(v)} \hat{f}^2_a = \mathrm{E}_{x,y\sim\{0,1\}^i, u\sim\{0,1\}^{n-i}}[f(xu)f(yu)\chi_v(x+y)]$$

which using the relation $\chi_v(x+y) = \chi_v(x)\chi_v(y)$ we can rewrite as

$$\sum_{a \in FIX(v)} \hat{f}^2_a = \mathrm{E}[(f(xu)\chi_v(x))(f(yu)\chi_v(y))] = \mathrm{E}_u[\mathrm{E}_x[f(xu)\chi_v(x)]^2] = \mathrm{E}_u[\hat{f}^{(u)2}_v],$$

where $f^{(u)}$ is the function obtained by restricting the inputs outside V to u. By linearity of expectation,

$$\sum_{(a,v):\, a \in FIX(v), |v|>k} \hat{f}^2_a = \mathrm{E}_u \sum_{v:\, |v|>k} \hat{f}^{(u)2}_v.$$

The left hand side now ranges over those a such that $|a_V| > k$, where a_V is the restriction of A on V. For the right hand side, notice that for a for a fixed u, all the coefficients $\hat{f}^{(u)}$ vanish if $f^{(u)}$ has decision tree at most k; otherwise, $\sum_v \hat{f}^{(u)2}_v = 1$ since $f^{(u)}$ is a $\{1,-1\}$ valued function. In any case, we get that

$$\mathrm{E}_u \sum_{a:\, |a_V|>k} \hat{f}^{(u)2}_v \leq \Pr_u[f^{(u)} \text{ does not have decision tree depth } k]$$

and so

$$\sum_{a:\, |a_V|>k} \hat{f}^2_a \leq \Pr_u[f^{(u)} \text{ does not have decision tree depth } k].$$

Now let V be a random subset of $[n]$ where each coordinate is included independently with probability p. Averaging over V, we get

$$\mathrm{E}_V \sum_{a:\, |a_V|>k} \hat{f}^2_a \leq \mathrm{E}_V \Pr_{u\sim[n]-V}[f^{(u)} \text{ does not have decision tree depth } k]$$

Now the right hand side is exactly the probability that f^ρ does not have decision tree depth k for a p-random restriction ρ, which is at most $s2^{-k}$ by Theorem 4. For the left hand side by linearity of expectation we can write

$$\mathrm{E}_V \sum_{a:\, |a_V|>k} \hat{f}^2_a = \sum_{a\in\{0,1\}^n} \hat{f}^2_a \Pr_V[|a_V| > k].$$

Then the expected weight of a_V is $p|a|$. By Chebyshev's inequality, the weight of a_V is at least $p|a| - \lambda\sqrt{p|a|}$ with probability $1 - 1/\lambda^2$. If $p|a| > 4$, we get that $|a_V| \geq p|a|/2$ with probability at least $1/2$ and so

$$\sum_{a \in \{0,1\}^n} \hat{f}_a^2 \Pr_V[|a_V| > k] > \sum_{a\colon |a| > 2k/p} \hat{f}_a^2 (1/2) \geq 1/2 \sum_{a\colon |a| > 2k/p} \hat{f}_a^2.$$

To conclude, as long as $k > 2$, we have that

$$\sum_{a\colon |a| > 2k/p} \hat{f}_a^2 \leq 2s \cdot 2^{-k}$$

and since $p = 1/(10k)^{d-1}$, we derive Theorem 1.

3 Influences and pseudorandom functions

The *influence* of the i-th input on a boolean function $f\colon \{0,1\}^n \to \{1,-1\}$ is the probability that flipping the value of the i-th input changes the value of f when the other inputs are chosen at random. We can write this as

$$\mathrm{I}_i[f] = \Pr_{x \sim \{0,1\}^n}[f(x) \neq f(x + e_i)]$$

where e_i is the unit vector in coordinate i.

There is a nice and useful formula for influence in terms of the Fourier transform of f. To derive this formula, we want to convert probabilities into expectations. One nice way to do this calculation is to look at the *derivative* function

$$D_i f(x) = \frac{f(x) - f(x + e_i)}{2}$$

This derivative takes value 0 if x does not contribute to the influence, and -1 or 1 if it does, so we can write

$$I_i[f] = \mathrm{E}_{x \sim \{0,1\}^n}[D_i f(x)^2] = \sum_{a \in \{0,1\}^n} \widehat{D_i f}_a^2$$

by Parseval's identity. So we just need to calculate the Fourier coefficients of $D_i f$:

$$\begin{aligned}
\widehat{D_i f}_a &= \mathrm{E}\left[\frac{f(x) - f(x + e_i)}{2} \cdot \chi_a(x)\right] \\
&= \frac{1}{2}\big(\mathrm{E}[f(x)\chi_a(x)] - \mathrm{E}[f(x + e_i)\chi_a(x)]\big) \\
&= \frac{1}{2}\big(\mathrm{E}[f(x)\chi_a(x)] - \mathrm{E}[f(x)\chi_a(x + e_i)]\big) \\
&= \frac{1}{2}\hat{f}_a(1 - \chi_a(e_i)) \\
&= \hat{f}_a \frac{1 - (-1)^{a_i}}{2}
\end{aligned}$$

Notice that this is $\hat{f}_a$ if $a_i = 1$ and 0 otherwise. So we have

$$\mathrm{I}_i[f] = \sum_{a\colon a_i = 1} \hat{f}_a^2.$$

The *total influence* $\mathrm{I}[f]$ of a boolean function f is the sum of the influences across coordinates. Its value is between 1 and n. Its combinatorial meaning follows by averaging over the definition of influence:

$$\mathrm{I}[f] = \mathrm{I}_1[f] + \cdots + \mathrm{I}_n[f] = \mathrm{E}_x[\text{number of } i \in [n] \text{ such that } f(x) \neq f(x + e_i)].$$

Using the fourier formula for influences, we get

$$\mathrm{I}[f] = \sum_{i=1}^{n} \sum_{a\colon a_i=1} \hat{f}_a^2 = \sum_{a \in \{0,1\}^n} |a| \hat{f}_a^2.$$

Notice that the more weight f has on its large fourier coefficients, the larger its influence is. Contrast this with Theorem 1, which says that shallow circuits cannot have much weight on their large fourier coefficients. This leads to the conclusion that functions computed by not too large shallow circuits must have small influence.

Proposition 5. *If $f\colon \{0,1\}^n \to \{1,-1\}$ is computable by a size s, depth d and-or circuit then $\mathrm{I}[f] = (O(\log ns))^d$.*

Proof. We write

$$\begin{aligned}
\mathrm{I}[f] &= \sum_{a\colon |a| \leq t} |a| \hat{f}_a^2 + \sum_{a\colon |a| > t} |a| \hat{f}_a^2 \\
&\leq t \cdot \sum_{a\colon |a| \leq t} \hat{f}_a^2 + n \cdot \sum_{a\colon |a| > t} \hat{f}_a^2 \\
&\leq t + 2ns \cdot 2^{-\Omega(t^{1/d})}
\end{aligned}$$

using Parseval's identity and Theorem 1. This inequality is valid for every t, so if we choose $t = (K \log ns)^d$ for a sufficiently large constant K we get the desired conclusion. □

Total influence is often a lot easier to calculate than the fourier spectrum, so Proposition 5 is a useful tool for showing certain functions are not computable by shallow circuits. You will work out some examples in the homework.

A *pseudorandom function family* is a distribution $\mathcal{F}$ over functions $\{f\colon \{0,1\}^n \to \{0,1\}\}$ such that a (bounded) adversary that is allowed to query f at arbitrary points cannot distinguish a function chosen from $\mathcal{F}$ from a truly random function. Such functions have applications in cryptography.

Using Proposition 5 we can argue that pseudorandom functions cannot be computed by shallow circuits. For consider the following test T for distinguishing a random function from a pseudorandom one: Choose a random input x, a random coordinate i, and output "pseudorandom" if $f(x) = f(x + e_i)$ and "random" otherwise. If f is computable by a size s, depth d circuit, by Proposition 5 T will output "pseudorandom" with probability $1 - O(\log ns)^d/n$. In contrast, since $x \neq x + e_i$, if f is truly random then T will output "pseudorandom" only with probability $1/2$. Repeating this test a few times allows us to distinguish size s, depth d functions from truly random ones with high confidence.