

In Lecture 1 we saw examples of functions that are hard to compute by small decision trees. Today we will study the complexity of two more general models of computations: Formulas in disjunctive normal form (DNFs) and small depth circuits.

One seemingly powerful method for proving size lower bounds for decision trees that we saw was the method of random restrictions. We showed that a small decision tree becomes shallow with high probability after a random restriction (while functions like PARITY and MAJORITY don't). This property of random restrictions generalizes to DNFs. It goes by the name of "the Switching Lemma".

1 The switching lemma

In the last lecture we defined a δ -random restriction of the variables x_1, \dots, x_n (taking $\{0, 1\}$ values) to be an random assignment where each coordinate is set *independently* to \star with probability δ and to 0 and 1 with probability $(1 - \delta)/2$ each. Under this definition, the number of stars in a random restriction is a random variable, although one that is heavily concentrated around its mean value of δn (as long as δn is not too small). For today's lecture it will be convenient to slightly modify this definition so that the number of stars in the restriction always equals exactly $u = \delta n$. This will simplify the proof of our switching lemma but make little difference in its applications.

Let n be the number of variables and u be a number between 0 and n . A u -restriction is an assignment $\rho \in \{0, 1, \star\}^n$ in which exactly u of the n coordinates are stars. A u -random restriction is a random assignment chosen uniformly from the set of all u -restrictions. It can be generated like this: First, set exactly u out of the n coordinates to \star uniformly at random. Then set each of the remaining $n - u$ coordinates independently to 0 or 1 with equal probabilities. (The number of zeros and ones may not be the same.)

The *width* of a DNF is the width of its widest clause, where the width of a clause is the number of literals in it. Width measures the amount of processing that the AND gates in the DNF have to perform in parallel. For example, the width of the following DNF ϕ is 3. (A bar denotes negation and concatenation denotes AND.)

$$\phi = \bar{x}_1 x_3 \text{ OR } x_1 \bar{x}_3 x_5 \text{ OR } x_2 x_3 \text{ OR } \bar{x}_3 x_6 \text{ OR } x_4 x_6. \quad (1)$$

Theorem 1 (The switching lemma). *Assume $u \leq n/2$. For every $f: \{0, 1\}^n \rightarrow \{0, 1\}$ computable by a DNF of width w and a u -random restriction ρ , the probability that $f|_\rho$ requires a decision tree of depth d is at most $(8wu/n)^d$.*

The switching lemma says that a DNF of sufficiently small width is likely to simplify after a random restriction. In particular, it implies that DNFs of small width cannot compute functions like *PARITY* or *MAJORITY* (you will show this in the homework). Let's understand why this should be the case. Suppose that some DNF of width w computed the *PARITY* function. Look at the first clause of this DNF. There must exist a fixing of the w variables in this clause that sets its value, and therefore the value of the whole DNF to 1. On the other hand, the value of *PARITY* is undetermined until all n of its variables are set, so *PARITY* requires DNF of width n .

Remarkably, the switching lemma makes no reference to the size of the DNF, although this is arguably the more natural complexity measure that we are interested in. The reason that Theorem 1 is also relevant for size is that any small DNF is likely to become narrow after a random restriction:

Claim 2. Assume $w \leq n/4$. For every $f: \{0,1\}^n \rightarrow \{0,1\}$ that has DNF size s and a $n/2$ -random restriction ρ , the probability that $f|_\rho$ requires a DNF of width w is at most $s \cdot (7/8)^w$.

Before we prove this simple claim, let's introduce some notation that will be useful later. Given a DNF ϕ and a restriction ρ , the restricted DNF $\phi|_\rho$ is the DNF obtained by substituting the partial assignment ρ into ϕ and performing the following simplifications: (1) If any clause evaluates to 1, set $\phi|_\rho$ to 1; (2) if not, eliminate from $\phi|_\rho$ all the clauses that evaluate to 0. For example, for the above DNF ϕ and $\rho = 11*0**$,

$$\phi|_\rho = \overline{x_3}x_5 \text{ OR } x_3 \text{ OR } \overline{x_3}x_6. \quad (2)$$

This definition is purely syntactic, unlike our definition of $f|_\rho$ for a *function* f from last lecture, which is semantic. It could be the case, for example, that given a DNF ϕ , the restriction ρ simplifies the function computed by ϕ to a constant, yet the *formula* for $\phi|_\rho$ is not a constant. Nevertheless, if the DNF ϕ computes function f , then the DNF $\phi|_\rho$ will certainly compute the function $f|_\rho$.

Proof of Claim 2. Let ϕ be a size s DNF for f . We show that the probability that $\phi|_\rho$ has a clause of width w or more is at most $s \cdot (7/8)^w$. Each surviving clause in $\phi|_\rho$ can only be narrower than the corresponding clause in ϕ , so it is enough to show that the probability that some clause of width w or more survives ρ is at most $s \cdot (7/8)^w$.

The probability that any given such clause survives ρ is at most $(7/8)^w$: Each literal is killed with probability $1/4$. If the assignments were independent, the probability of survival would have been exactly $(3/4)^w$. Under our model for ρ , the probability that the $(j+1)$ -st variable is killed given that all previous variables have survived is at least $(n/2 - j)/n \cdot 1/2$, the product of the probability that it gets a non-starred value and the (conditional) probability that it gets the "killer" value. Under our assumptions this is at most $1/8$, so the probability that at least one variable doesn't survive is at least $(7/8)^w$. To finish the proof, we take a union bound over all clauses of width at most w . \square

From Theorem 1 and Claim 2, it follows easily that *PARITY* requires DNFs of exponential size. To see this assume that there is a size s DNF for *PARITY* on n variables. By Claim 2, there exists a $n/2$ -restriction ρ and a DNF of width $w = 6 \log s$ for *PARITY* $|_\rho$. By Theorem 1 with $u = n/50 \log s$, there is a $O(n/\log s)$ -restriction ρ' so that the function obtained by applying restriction ρ' to *PARITY* $|_\rho$ is constant (i.e. it has decision tree depth zero). But if $s < 2^{n/50}$, the composition $\rho\rho'$ of the two restrictions still has an unrestricted variable, so *PARITY* $|_{\rho\rho'}$ cannot be a constant function.

2 Proof of the switching lemma

To prove the switching lemma, we will upper bound the *number* of "bad" restrictions ρ for which $f|_\rho$ does not have a decision tree of depth less than d . The counting is done in an indirect way: We will show that each such u -restriction ρ can be uniquely described by a $(u-d)$ -restriction κ plus a small amount of "auxiliary information". Ignoring the auxiliary information for a moment, let us see how this type of argument goes towards proving Theorem 1. The number of u -restrictions is $\binom{n}{u} 2^{n-u}$, and if every bad u -restriction uniquely maps to some $(u-d)$ -restriction, the probability that a random u -restriction is bad would be at most

$$\binom{n}{u-d} 2^{n-(u-d)} / \binom{n}{u} 2^{n-u} \quad \text{which is at most} \quad \left(\frac{u}{n-u}\right)^d 2^d \leq (4u/n)^d.$$

which is better than what we promised.

Let's think of the map as a game between an Alice who wants to communicate a restriction ρ to Bob by a message shorter than ρ itself. We assume that both Alice and Bob know the formula ϕ .

An example To be concrete, let's take again ϕ to be the DNF (1) and $\rho = 11\star 0\star\star$. Then $\phi|_\rho$ is the DNF (2). Alice takes the first clause of $\phi|_\rho$, in this case the clause \bar{x}_3x_5 . She then finds the unique partial assignment α that makes \bar{x}_3x_5 true, namely $x_3 = 0$ and $x_5 = 1$. The restriction κ is obtained by extending ρ with the assignment α : Alice sends $\kappa = 11001\star$ to Bob.

Can Bob now recover ρ from κ ? The issue is that Bob does not know which of the 0s and 1s in κ came from the assignment α and which were originally present in ρ . But Bob can try to reverse-engineer Alice's steps. Applying the restriction κ to the ϕ , Bob sees that the first clause \bar{x}_1x_3 vanishes under κ , so it is not the clause set by Alice. However, the next clause $x_1\bar{x}_3x_5$ evaluates to 1 under κ , so Bob knows that the partial assignment α only involves some of the variables x_1, x_3, x_5 . But which ones of the three were set by ρ and which were set by α ? To allow Bob to determine this, Alice sends as additional information the identities of the variables among x_1, x_3 , and x_5 that were starred in ρ and set in α .

To summarize, Alice encodes the restriction $\rho = 11\star 0\star\star$ by the pair (κ, aux) , where κ is the restriction $11001\star$ and aux describes variables x_3 and x_5 within the clause $x_1\bar{x}_3x_5$. One way to describe these two variables is to specify their set of indices, namely $aux = 35$. A more succinct encoding is to give the set of indices *within the clause* $x_1\bar{x}_3x_5$, namely $aux = 23$, referring to the fact that x_3 and x_5 are the second and third variable within this clause. For the proof of the switching lemma it is essential that Alice and Bob make use of the latter encoding. Upon receiving this information, to recover ρ , Bob looks for the first clause of ϕ that is set to 1 by κ and replaces the variables indexed by aux with stars, where the indexing refers to the order in which the variables appear within this clause.

So far we have shown how to encode (for this specific width 3 DNF ϕ) our 3-restriction ρ by a 1-restriction κ together with $3 - 1 = 2$ entries of auxiliary information, each taking values between 1 and w . Extrapolating from this example, the encoding represents a u -restriction ρ by a $(u - d)$ -restriction κ and a string aux with d symbols in the set $\{1, \dots, w\}$, where d is the width of the first surviving clause of $\phi|_\rho$. The probability that a clause of width d or more survives the restriction is therefore at most

$$\frac{\text{number of } (u - d)\text{-restrictions}}{\text{number of } u\text{-restrictions}} \cdot w^d \leq \left(\frac{u}{n - u}\right)^d 2^d w^d = (2wu/(n - u))^d$$

which has the form that we want. However, d here refers to the width of the first surviving clause, while we want it to be the depth of the best decision tree for $f|_\rho$.

To make further progress, a natural idea is to fix more variables in ρ by applying the encoding recursively to κ . This appears to be a non-starter, as $\phi|_\kappa = 1$. What Alice and Bob will do instead is agree to extend ρ to another restriction ρ' which also fixes the variables x_3 and x_5 , but to different values than in κ , so that the clause \bar{x}_3x_5 does not become true under ρ' . There are three such possible assignments for x_3x_5 : 00, 01, and 11. Among these three, which one should Alice choose? Setting x_3 to 1 is not a good idea as it makes the second clause of $\phi|_\rho$ true, thereby impeding further progress. Alice picks the partial assignment β which sets x_3x_5 to 00 (it doesn't matter which of the remaining two in this case), extends ρ by β to obtain ρ' – in this case $\rho' = 11000\star$ – communicates ρ' to Bob succinctly by specifying the partial assignment 00, and then iteratively encodes ρ' by editing the relevant parts of κ and supplying the required auxiliary information.

Encoding and decoding In general, Alice will use the following procedure to encode her restriction ρ . We assume Alice and Bob share the formula ϕ , but the restriction ρ is only known to Alice.

Procedure *Enc*: On input $\rho \in \{0, 1, \star\}^n$,

Set $\kappa = \rho$, $\rho' = \rho$, and *aux* to the empty string.

Until $\phi|_{\rho'}$ is a constant repeat the following:

Let $c|_{\rho'}$ be the first clause of $\phi|_{\rho'}$ and c the corresponding clause of ϕ .

Let α be the partial assignment that sets $c|_{\rho'}$ to true.

Let β be the partial assignment to $c|_{\rho'}$

that maximizes the decision tree depth of the function $\phi_{\rho'\beta}$.

Extend κ by α and ρ' by β .

Append to the tail of *aux*

the list of variables set by α (and β) indexed by their position in c and their values in β

Append the symbol \cdot to *aux*.

Output the pair (κ, aux) .

Before stating the properties of this procedure, let us run thru our example ϕ from (1) and $\rho = 11\star 0\star\star$. In the first iteration, we obtain as described above

$$\phi|_{\rho'} = \overline{x_3}x_5 \text{ OR } x_3 \text{ OR } \overline{x_3}x_6, \quad c = x_1\overline{x_3}x_5 \quad c|_{\rho'} = \overline{x_3}x_5 \quad \alpha = 01 \quad \beta = 00.$$

At the end of this iteration, κ , ρ' , and *aux* become

$$\kappa = 11001\star \quad \rho' = 11000\star \quad \text{aux} = 2300\cdot$$

as the second and third variable in $c|_{\rho'}$ both take value 0 in ρ' . So in the next iteration, we have

$$\phi|_{\rho'} = x_6, \quad c = \overline{x_3}x_6 \quad c|_{\rho'} = x_6 \quad \alpha = 1 \quad \beta = 0.$$

At the end of this iteration, $\kappa = 110011$, $\rho' = 110000$ and *aux* = 2300·20·. Since $\phi|_{\rho'}$ is now a constant, *Enc*(ρ) terminates with the encoding

$$\text{Enc}(\rho) = (110011, 2300\cdot 20\cdot).$$

which Alice sends to Bob.

Bob recovers ρ from the pair (κ, aux) by applying Alice's steps in reverse:

Procedure *Dec*: On input (κ, aux) :

Let $\rho = \kappa$ and $\rho' = \kappa$.

Loop through the parts a of *aux* separated by dots:

Let c be the first clause of ϕ satisfied by ρ' .

Replace the entries indexed by a in ρ by stars.

Replace the same entries in ρ' by the corresponding assignment in a .

Output ρ .

Initially, $\rho = 110011$ and $\rho' = 110011$. Bob begins by finding the first clause of c that is set to 1 by ρ' , in this case $c = x_1\overline{x_3}x_5$, and then modifies ρ and ρ' by resetting the values of x_3 and x_5 as described in the first part of *aux* to obtain

$$\rho = 11\star 0\star 1 \quad \rho' = 110001.$$

In the next iteration, $c = \overline{x_3}x_6$. The second part of *aux* says that the second variable x_6 in this clause was set to 0 in ρ' , so we recover

$$\rho = 11\star 0\star\star \quad \text{and} \quad \rho' = 110000.$$

Properties of the encoding In general, one can prove the following claim:

Claim 3. *For every partial assignment ρ , $\text{Dec}(\text{Enc}(\rho)) = \rho$, and so Enc is an injective map.*

We now argue that if $\phi|_\rho$ has a large decision tree, then κ contains a lot fewer stars than ρ :

Claim 4. *If $\phi|_\rho$ requires decision tree depth $d > 0$, then κ has at least d fewer stars than ρ .*

Proof. We prove the claim now by strong induction on d . When $d = 0$ there is nothing to prove. Now assume it is true for all values smaller than d . The assignment β is chosen so as to maximize the decision tree depth of the function $\phi|_{\rho\beta}$. Since $d > 0$, $\phi|_\rho$ is not constant. If its first clause $c|_\rho$ contains v variables then we will show that decision tree depth of $\phi|_{\rho\beta}$ must be at least $d - v$. By inductive hypothesis, κ had at least $d - v$ fewer stars than $\rho\beta$, so it has at least d fewer stars than ρ .

Suppose, for contradiction, that for every partial assignment β to these variables, $\phi|_{\rho\beta}$ had a decision tree of depth less than $d - v$. We can then construct a decision tree for $\phi|_\rho$ of depth less than d : First query the v variables in $c|_\rho$. For the unique assignment that sets c to true output 1. For every other assignment β , run the decision tree for $\phi|_{\rho\beta}$. This is a decision tree for $\phi|_\rho$ of combined depth less than d . \square

By a small modification of the encoding procedure, we can in fact ensure that κ has precisely d fewer stars than ρ (assuming $\phi|_\rho$ requires decision tree depth d): Once exactly d starred variables in ρ are assigned values in κ , the encoding procedure halts. This does not affect the correctness of the decoding, so the encoding map remains injective.

To summarize, we have shown that every u -restriction ρ such that $\phi|_\rho$ requires decision tree depth d can be uniquely described by a $(u - d)$ -restriction κ together with some auxiliary information aux . If A is the set of all possible values for aux , it follows that

$$\begin{aligned} \Pr_\rho[\phi|_\rho \text{ requires decision tree depth } d] &\leq \frac{\text{number of } (u - d)\text{-restrictions } \kappa}{\text{number of } u\text{-restrictions } \rho} \cdot |A| \\ &= \frac{\binom{n}{u-d} 2^{n-(u-d)}}{\binom{n}{u} 2^{n-u}} \cdot |A| \\ &\leq (2u/n)^d |A| \end{aligned}$$

so all that remains to do is to bound the size of A . To specify aux , we need to give d symbols taking values in the set $\{1, \dots, w\}$ (describing variables within a clause), an additional d symbols taking 0, 1 values (assignments to these variables), some of which are followed by a dot. If we extend the assignment alphabet from $\{0, 1\}$ to $\{0, 1, 0\cdot, 1\cdot\}$ to account for the dots, we conclude that aux can take at most $w^d \cdot 4^d$ possible values, so $|A| \leq (4w)^d$. The desired probability is then at most $(4wu/n)^d$, which is what we needed to prove.

3 Small depth circuits

An *AND/OR circuit* of unbounded fan-in is a directed acyclic graph¹ in which the nodes are assigned the following labels: Each source node is labeled by exactly one of the literals $x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}$, while the other nodes are labeled by AND or OR. In addition, each of the m sinks gets a unique label among y_1, \dots, y_m . (If there is only one sink we might not bother assigning it a label.)

¹It is curious that circuits should be acyclic, but this is the standard terminology in computational complexity, probably tracing its roots back to the origins of computing devices in electrical engineering.

A circuit computes a function from $\{0, 1\}^n$ to $\{0, 1\}^m$ in a natural way: The inputs (x_1, \dots, x_n) and their negations $(\bar{x}_1, \dots, \bar{x}_n)$ are plugged in at the source, and then the values of the internal nodes are computed until the outputs (y_1, \dots, y_m) are found.

The *size* of the circuit is the number of nodes, excluding the source nodes. Its *depth* is the length of the longest path from source to sink. Size represents the length of the program, or number of gates in a piece of hardware modeled by the circuit. Depth is the time it takes to evaluate the circuit in parallel (assuming the gates at distance t from the input are evaluated at time step t).

ANDs and ORs of literals are circuits of depth 1, while a DNF is a circuit of depth 2. A natural question to ask is how the depth of circuits affects their computational power. When the depth is small, the switching lemma is a powerful tool for analyzing such circuits.

Theorem 5. *If $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is computable by a circuit of size s and depth d and ρ is a $n/(K \log s)^{d-1}$ -random restriction then the probability that $f|_\rho$ requires a decision tree of depth t is at most $d/s + 2^{-t}$, where K is some constant.*

The error probability can be made to vanish faster by choosing different constants, but let's stick to this statement for concreteness. To prove the theorem we will apply the switching lemma iteratively, arguing that at each step the bottom layer of gates vanishes with high probability over the choice of random restriction.

Proof. By the same proof as in Claim 2, if ρ_0 is an $n_0 = n/2$ random restriction the probability that any of the gates at level 1 has fan-in more than $20 \log s$ is at most $1/s$. Assuming this is not the case, namely all level 1 gates have fan-in at most $20 \log s$ after applying ρ_0 , all expressions computed at level 2 are DNFs or CNFs (complements of DNFs) of width at most $w = 20 \log s$.

By Theorem 1, if we set $n_1 = n_0/200 \log s$, the probability that any of the DNFs or CNFs does not simplify to a decision tree of depth at most w under a n_1 -random restriction ρ_1 is at most $s \cdot (8wn_1/n_0)^{20 \log s} \leq 1/s$. Assuming all level 2 functions do simplify, the level 3 gates of the circuit are now ORs or ANDs of decision trees of depth at most w . By Theorem 6 from Lecture 1, each such decision tree can be expanded as a DNF, or by an analogous argument a CNF, of width w . The top gate of these DNFs or CNFs can then be merged with a level 3 gate of the same type, thereby reducing the depth of the circuit by 1.

Let ρ be the restriction obtained by applying $\rho_0, \rho_1, \dots, \rho_{d-1}$ in this order. If n_i is the input size of $f|_{\rho_0 \dots \rho_i}$ for $i \geq 1$ then ρ_{i+1} is an n_{i+1} -random restriction with $n_{i+1} = n_i/200 \log s$. At each step except for the last one, the probability that the depth of the circuit does not decrease is at most $1/s$. In the last step, we apply Theorem 1 with depth parameter t . By a union bound, with probability at least $1 - d/s - 2^{-t}$ the depth reduces at each step. In this case $f|_\rho$ is a decision tree of depth w .

The number of unrestricted variables in ρ is $n_0/(200 \log s)^{d-1} = \frac{1}{2}n/(200 \log s)^{d-1}$ as desired. \square

Applying Theorem 5 to the *PARITY* function, we conclude the following:

Theorem 6. *If a circuit that computes *PARITY* on n bits has depth d , its size must be $2^{\Omega(n^{1/(d-1)})}$.*

Proof. If $n > (K \log s)^{d-1}$ in the notation of Theorem 5, then $\text{PARITY}|_\rho$ is not a constant function, but $f|_\rho$ has a decision tree of depth 0 (and is therefore constant) with some probability for any f that admits a circuit of size s and depth d . Therefore *PARITY* cannot be computed by such circuits. Choosing the smallest value of n gives $s = 2^{\Omega(n^{1/(d-1)})}$. \square

This bound for the *PARITY* function is tight as circuits of depth d and size $2^{O(n^{1/(d-1)})}$ for *PARITY* on n bits do exist. A slightly weaker lower bound can be proved for the *MAJORITY* function as well. You will investigate this in the homework.

All the lower bounds we have seen so far apply to the *PARITY* function as well as to the *MAJORITY* function. Which of the two is the harder one to compute? One way to make sense of this question is to augment our AND/OR circuit model with a new type of gate. Can small AND/OR/MAJORITY circuits compute *PARITY*? What about small AND/OR/PARITY circuits, can they compute *MAJORITY*?

It turns out that the answer to the first question is yes, while the answer to the second one is a no. *PARITY* on n bits can be computed by circuits with just MAJORITY gates of depth 3 and size $O(n)$. In contrast, Razborov showed that *MAJORITY* requires depth d AND/OR/PARITY circuits of size $2^{n^{\Omega(1/d)}}$. In this sense, *MAJORITY* is much harder to compute than *PARITY* in small depth.

References

The switching lemma was proved by Håstad. The proof presented here is due to Razborov. A weaker variant of the lemma was previously discovered by Furst, Saxe, and Sipser, who applied it to prove circuit lower bounds for the *PARITY* function. Intermediate improvements and other applications were given by Yao and Ajtai.