CMPT 407 - Complexity Theory: Lecture 11

Valentine Kabanets

July 3, 2017

1 k-SAT algorithm from Håstad's Switching Lemma

Earlier we saw how the Switching Lemma can be used to prove exponential lower bounds for AC^0 circuit size (against PARITY). However, the same Switching Lemma also has "algorithmic" applications: to computational learning [LMN93] as well as to SAT [IMP12]. Today we describe the latter application, implicit in [IMP12]; the randomized algorithm for k-SAT we describe is a base case of the more general algorithm given in [IMP12] that would decide the satisfiability of a given AC^0 circuit (not just a k-CNF). Recall that the k-SAT problem asks to decide if a given k-CNF (a conjunction of clauses, where each clause is of size at most k) is satisfiable.

Theorem 1 ([IMP12]). *There is a zero-error randomized algorithm for k-SAT running in time $2^{n(1-\epsilon/k)}$, for some constant $\epsilon > 0$. Moreover, there is a zero-error randomized algorithm, running in the same time, that counts the number of satisfying assignments of a given k-SAT instance.*

The algorithm for k-SAT will depend on the general version of the Switching Lemma, which we state next.

1.1 Canonical Decision Trees

A *decision tree* is an algorithm to compute a given boolean function $f(x_1, \ldots, x_n)$ by querying its variables in a certain order, until the value of the function on a given input is determined. More precisely, a decision tree is a binary tree, with internal nodes labeled by variables $(x_1, \ldots, x_n)$, the leaves labeled by constants 0 or 1, and each internal node having two children: the left and the right, with the left edge labeled by 0 and the right with 1. The meaning is: to evaluate f on a given input $a_1, \ldots, a_n$, we start at the root of the decision tree, say variable x_3, and query that variable. If its value (a_3) is 0, we go to the left child, and otherwise to the right child, and continue this way until we reach a leaf, whereupon we output the label of the leaf as the final value of $f(a_1, \ldots, a_n)$. The *depth* of the decision tree is the length of a longest branch of the tree.

Given a k-CNF $\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$, with m clauses C_i, $1 \leq i \leq m$, on at most k literals each, we define the *canonical decision tree (CDT)* of ϕ as follows:

- Form a complete decision tree that queries all variables in clause C_1 (in some canonical order); this tree will have height k if clause C_1 has k literals in it. Each branch of this decision tree determines a partial assignment (assigning the variables of C_1).
- For each branch of this decision tree, we plug in this partial assignment into our k-CNF, and simplify it (by removing all true clauses, and checking of any of the clauses becomes empty (unsatisfied)). After the simplification, if the resulting new k-CNF is non-empty, we take the first clause C_i, $i > 1$, still present in the k-CNF, and extend our current branch by the full decision tree querying the (remaining) variables of this clause C_i.
- We continue recursively, until the value of the k-CNF gets determined (to be 0 or 1), at which point we end with a leaf with the corresponding label.

Håstad's Switching Lemma says that the depth of the CDT of a given k-CNF is likely to be small.

Lemma 2 (Håstad's Switching Lemma)**.** *For any $k > 0$, let $\phi(x_1, \ldots, x_n)$ be any k-CNF. For any $0 < p < 1$, and any $t \geq 0$, let $\rho \in R_p$ be a p-random restriction (keeping each variable free with probability p, and assigning it a constant 0 or 1 with erqual probability otherwise). We have*

$$\mathbf{Pr}_{\rho \in R_p}[\textit{the depth of CDT}(\phi|_\rho) > t] < (5pk)^t.$$

Note that every given decision tree of depth at most t can be written as a t-CNF (and as a t-DNF): we simply take the OR over all branches leading to a 1-leaf, of the AND of the corresponding literals on the branch. Thus the previous version of the Switching Lemma (used before) is a special case of the more general version given now.

Also note that, given a restriction $\rho \in R_p$, there is an efficient (polytime) algorithm that will construct a CDT for $\phi|_\rho$. The Switching Lemma says that, with high probability, this particular CDT will have small depth!

1.2 k-SAT algorithm

We first describe a randomized algorithm for k-SAT that is always correct, but has *expected* runtime $T = 2^{n(1-\epsilon/k)}$, for some constant $\epsilon > 0$. Such an algorithm can be easily converted to a randomized zero-error algorithm of about the same worst-case runtime (by simply running the original algorithm for $4T$ steps, outputting the answer if the first algorithm terminated within that time, and outputting the '?' otherwise).

For a parameter $0 < p < 1$ to be determined, we define the following algorithm.

On a given k-CNF $\phi(x_1, \ldots, x_n)$,

1. pick a random subset $S \subseteq \{1, 2, \ldots, n\}$, by deciding for each $1 \leq i \leq n$, independently at random, to place i into S with probability p. (Note that the expected size of the set S is np.)

2. Enumerate over all partial assignments a to the variables x_j for $j \notin S$, and construct the CDT for $\phi|_a$ for each such partial assignment.
3. Output 'Yes, satisfiable', if at least one of the constructed CDTs has at least one branch leading to a 1-leaf. Otherwise, output 'No, unsatisfiable'.

The intuition: a random set S chosen in step 1 of the algorithm, plus a given partial assignment a (for a random a) chosen in step 2, can be viewed as a random assignment $\rho \in R_p$, where we choose ρ in two steps: first a set S of free variables, and then assignment a to the variables outside of S that need to be fixed to constants. By the Switching Lemma, for a random p-restriction ρ, the resulting CDT is likely to have small depth and hence be small (exponential in the depth). In particular, we expect that for many of the partial assignments a, the CDT for $\phi|_a$ will be of small size, which will give us the required savings for the runtime of our SAT algorithm, in expectation. We provide more details next.

1.3 Analysis

First note that the described algorithm is always correct! Next we analyze its expected runtime. For simplicity, we will ignore polynomial factors in our analysis below. The *expected runtime* of the k-SAT algorithm described above on a given k-CNF $\phi(x_1, \ldots, x_n)$ is:

$$\mathbf{Exp}_S \left[\sum_{y \in \{0,1\}^{n-|S|}} |CDT(\phi|_{S,y})| \right],$$

where S is a random set chosen in Step 1 of the algorithm, y is a particular partial assignment to the variables outside of S, and $|CDT(\psi)|$ is the size of the CDT for a formula ψ. By rewriting the expectation over S as a summation, and rewriting the summation over y's as an expectation over uniform y (by adding an extra normalization factor $2^{n-|S|}$), we can equivalently write the expected runtime as follows:

$$\sum_S \mathbf{Pr}[S] \cdot 2^{n-|S|} \cdot \mathbf{Exp}_{y \in \{0,1\}^{n-|S|}} \left[|CDT(\phi|_{S,y})| \right].$$

Next, we split the summation over S into two sums: over those S with size $|S| \geq pn/2$, and those S with $|S| < pn/2$:

$$\sum_{S:|S| \geq (pn)/2} \mathbf{Pr}[S] \cdot 2^{n-|S|} \cdot \mathbf{Exp}_{y \in \{0,1\}^{n-|S|}} \left[|CDT(\phi|_{S,y})| \right] \tag{1}$$

$$+ \sum_{S:|S| < (pn)/2} \mathbf{Pr}[S] \cdot 2^{n-|S|} \cdot \mathbf{Exp}_{y \in \{0,1\}^{n-|S|}} \left[|CDT(\phi|_{S,y})| \right]. \tag{2}$$

By the Chernoff bound, $\mathbf{Pr}_S[|S| < pn/2] < 2^{-pn/12}$. Also, $|CDT(\phi|_{S,y})| \leq 2^{|S|}$, for every

S and y. Using these two facts, we can upper-bound the summation in Equation (2) by

$$\begin{aligned}\sum_{S:|S|<(pn)/2} \mathbf{Pr}[S] \cdot 2^n &= 2^n \cdot \mathbf{Pr}_S[|S| < (pn)/2] \\ &< 2^n \cdot 2^{-pn/12} \\ &= 2^{n(1-p/12)}.\end{aligned}$$

Next, we upper-bound the summation in Equation (1):

$$\begin{aligned}&\sum_{S:|S|\geq(pn)/2} \mathbf{Pr}[S] \cdot 2^{n-|S|} \cdot \mathbf{Exp}_{y\in\{0,1\}^{n-|S|}}\left[|CDT(\phi|_{S,y})|\right] \\ &\quad\leq \sum_{S:|S|\geq(pn)/2} \mathbf{Pr}[S] \cdot 2^{n-(pn)/2} \cdot \mathbf{Exp}_{y\in\{0,1\}^{n-|S|}}\left[|CDT(\phi|_{S,y})|\right] \\ &\quad\leq 2^{n(1-p/2)} \cdot \sum_{S} \mathbf{Pr}[S] \cdot \mathbf{Exp}_{y\in\{0,1\}^{n-|S|}}\left[|CDT(\phi|_{S,y})|\right] \\ &\quad= 2^{n(1-p/2)} \cdot \mathbf{Exp}_S\left[\mathbf{Exp}_y\left[|CDT(\phi|_{S,y})|\right]\right] \\ &\quad= 2^{n(1-p/2)} \cdot \mathbf{Exp}_{\rho\in R_p}\left[|CDT(\phi|_\rho)|\right],\end{aligned}$$

where in the third line we use the summation over all possible S (rather than only those of large size).

Next, we upper-bound the size of the CDT by $2^{\text{depth of CDT}}$. Also, we use the fact (proved in earlier lectures) that, for an integer random variable $X \geq 0$, we have

$$\mathbf{Exp}[X] = \sum_{i=0}^{\infty} \mathbf{Pr}[X > i].$$

Applying these two facts to the random variable $|CDT(\phi|_\rho)|$, we continue upper-bounding Equation (1) as follows:

$$\begin{aligned}2^{n(1-p/2)} \cdot \mathbf{Exp}_{\rho\in R_p}\left[|CDT(\phi|_\rho)|\right] &\leq 2^{n(1-p/2)} \cdot \sum_{i>0} \mathbf{Pr}_{\rho\in R_p}[\text{depth of } CDT(\phi|_\rho) \ > \log_2 i] \\ &< 2^{n(1-p/2)} \cdot \sum_{i>0} (5pk)^{\log_2 i},\end{aligned}$$

by the Switching Lemma. Finally, choose $p = 1/(20k)$. Then $5pk = (1/4)$, and so $(5pk)^{\log_2 i} = 1/i^2$. It is well-known that $\sum_{i>0} 1/i^2 \leq O(1)$ (in fact, is equal to $\pi^2/6$, but the exact constant is not important for us here). Thus, we conclude that the summation in Equation (1) is at most $O(1) \cdot 2^{n(1-p/2)}$.

Overall, the entire expected runtime is at most

$$O(1) \cdot 2^{n(1-p/2)} + 2^{n(1-p/12)} \leq O(1) \cdot 2^{n(1-p/12)} \leq O(1) \cdot 2^{n(1-1/(240k))},$$

as promised (for $\epsilon = 1/240$). This concludes the proof of Theorem 1 for the satisfiability question.

To count the number of satisfying assignments, we use the same randomized algorithm as before, except when we enumerate over all possible CDTs (over all possible partial assignments a), we *count* the number of satisfying assignments for each CDT: each branch of depth ℓ leading to a 1-leaf contributes $2^{n-\ell}$ to the sum. As CDT is a decision tree, it is easy to see that different branches determine *disjoint* subsets of partial assignments, so the counting over different branches counts every satisfying assignment exactly once!

References

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for ac^0. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972. SIAM, 2012.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.