CS265, Fall 2022

Class 11: Agenda and Questions

1 Announcements

- HW5 due Friday!

2 Recap/Questions?

Any questions from the minilectures and/or the quiz (second moment method and LLL)?

3 Practice with the LLL

Recall the *k-SAT* problem. There are n variables $x_1, \ldots, x_n$. We consider clauses that looks like $(x_{i_1} \vee x_{i_2} \vee \overline{x_{i_3}} \vee \cdots \vee x_{i_k})$; that is, a clause is the OR of k literals. **For today, assume that each clause has k distinct variables that appear in it.** We have a formula φ that is the AND of m clauses. We would like to know: is φ satisfiable? That is, is there a way to assign values to the variables $x_1, x_2, \ldots$ so that φ evaluates to TRUE?

Group Work

Suppose that each variable x_i is in at most t clauses, for some parameter t that will depend on k and that you'll work out in this problem. Apply the LLL to get a statement like the following:

Suppose that each variable is in at most t clauses of φ. Then φ is satisfiable.

(You should try to get t to be as large as possible. It's not hard to see that the statement above is true if, say, $t = 1$, but you should get a value of t that grows with k.)

***Hint**: Recall that to apply the LLL, you need to define a probability distribution and a set of "bad" events. We set up this example in the minilecture video, we just didn't work out the conclusion. In the set-up of the video, we considered the probability distribution to correspond to assigning TRUE/FALSE to each variable $x_1, \ldots, x_n$ independently with probability $1/2$ each, and we defined the bad event A_i to be the event that clause i is not satisfied.*

Group Work: Solutions

We claim that if each variable is in at most $t \leq 2^{k-2}k$ clauses, then the formula is satisfiable. To see this, consider a uniformly random assignment to $x_1, \ldots x_n$ (ie setting each

x_i to be TRUE/FALSE independently with probability 1/2). Define events $A_1, \ldots, A_m$ where A_i is the indicator random variable of the ith clause NOT being satisfied. For a clause with k variables to not be satisfied, all k variables must take the "bad" assignment, and hence:

$$\Pr[A_i] = 1/2^k.$$

To apply the LLL, we now need to reason about the dependencies. To that end, we claim that A_i is mutually independent of the set of clauses whose variable are disjoint from the variables in clause i, namely the set

$$S_i = \{A_j : vbl(clause_i) \cap vbl(clause_j) = \emptyset\}.$$

Indeed, no matter the assignment to variables that occur in the clauses in S_i, since none appear in the ith clause, they can't alter the probability of A_i. Now we simply count up the number of events not in the set S_i: namely

$$[\#j \text{ such that } vbl(clause_i) \cap vbl(clause_j) \neq \emptyset] \leq kt,$$

since there are k variables in clause i, and each of them is in at most t other clauses.

To conclude, each A_i is mutually independent of all but $d = kt$ other events, and hence by the LLL with $d = kt$ and $p = 2^{-k}$, we have that $\Pr[\cap_i not(A_i] \geq (1-2p)^m > 0$ provided $dp \leq 1/4$, hence we want $kt \cdot 2^{-k} \leq 1/4$, which implies that want $t \leq 2^{k-2}/k$.

To put some concrete numbers in here, if $k = 10$, then as long as each variable appears in at most $2^{10-8}/8 = 25.6$ clauses, then the formula is always satisfiable, no matter the number of variables of clauses!!! Of course, now the big question on your mind should be "*Its great that we know such formulas are satisfiable, but how do we FIND a satisfying assignment efficiently?*" We'll get to this in the next set of minilectures, on the "Constructive LLL"!!

3.1 More Practice with LLL and Mutual Independence

Here's an example where the mutual independence requirement is a bit trickier to think about. Consider a set of m equations over variables $x_1, \ldots, x_n$:

$$\begin{aligned}
\sum_{j=1}^{n} a_j^{(1)} x_j &\equiv b^{(1)} \mod 17 \\
\sum_{j=1}^{n} a_j^{(2)} x_j &\equiv b^{(2)} \mod 17 \\
&\vdots \\
\sum_{j=1}^{n} a_j^{(m)} x_j &\equiv b^{(m)} \mod 17
\end{aligned}$$

where:

- For all $j = 1, \ldots, n$ and all $r = 1, \ldots, m$, the coefficients $a_j^{(r)} \in \{0, 1, 2, \ldots, 16\}$ are not all zero; and
- for all $r = 1, \ldots, m$, $b^{(r)} \in \{0, 1, \ldots, 16\}$.

Suppose that each variable x_j appears in at most 4 of the m equations. (That is, for each j, $a_j^{(r)} = 0$ for all but four values of r.)

Group Work

With the setup above, prove that there exists an assignment to the variables such that *none* of the equations are satisfied.

***Hint**: Recall that because 17 is prime, for any $a \in \{1, \ldots, 16\}$ and any $b \in \{0, \ldots, 16\}$, the equation $ax \equiv b \mod 17$ has a unique solution for $x \in \{0, \ldots, 16\}$.*

***Hint**: It might be helpful to go back to the definition of mutual independence when arguing about the value of d when applying the LLL.*

Group Work: Solutions

Let's assign each x_i a random value in $\{0, 1, \ldots, 16\}$. We will define our bad events as follows. Let A_r be the event that the r'th equation is satisfied. The probability that this occurs is $1/17$. Indeed, by assumption there is some j so that $a_j^{(r)} \neq 0$. Conditioned on the values of x_i for $i \neq j$, the r'th equation reads:

$$a_j^{(r)} x_j \equiv b^{(r)} - \sum_{i \neq j} a_i^{(r)} x_i \mod 17$$

where we've moved everything deterministic (including the stuff we've conditioned on) to the right hand side of the equation. By the hint, there is exactly one value of $x_j \in \{0, \ldots, 16\}$ that will satisfy this equation, and so the probability that we hit that x_j is exactly $1/17$. Since this holds no matter what values we've conditioned on for the $x_i, i \neq j$, the probability that A_r occurs is also $1/17$.[a]

Now we need to determine the "d" parameter in the LLL. We claim that $d \leq 3$, in which case we have

$$dp = 3 \cdot \frac{1}{17} \leq 1/4$$

and we can apply the first version of the LLL from the lecture notes.

Let's first try the sort of argument we tried for the k-SAT example above. (This won't work!) Each equation contains at most n variables, and each variable occurs in at most 4 other variables, so there are at most $4n$ other equations that share any variable with the r'th equation. Clearly any equation that doesn't share any variables is independent,

so we can take $d = 4n$. That's correct, but this is not what we wanted! We'd get $dp = 3n/17$ which will be much larger than $1/4$ for any $n \geq 2$.

Instead, let's dig in a bit to the definition of mutual independence. For a given r, we want to show that there is some set $S \subseteq [m]$ of size at most 3 so that for any $J \subseteq [m] \setminus S$, $\Pr[A_r | \cap_{\ell \in J} A_\ell] = \Pr[A_r]$. Suppose that j is such that $c_j^{(r)} \neq 0$. (This exists by assumption). By assumption, there are at most three other equations so that x_j appears in them. Let S be this set of at most three r's. Now consider any set $J \subseteq [m] \setminus S$. Just as we did above, condition on all of the values of x_i other than x_j. (For example, set $x_i = y_i$ for some fixed y_i for all $i \neq j$. Then for all $\ell \in J$, A_ℓ is a deterministic event, since all of the variables that appear in the ℓ'th equation have already been fixed. Thus:

$$\Pr[A_r | \cap_{\ell \in J} A_\ell, x_i = y_i \forall i \neq j] = \Pr[A_r | x_i = y_i \forall i \neq j] = 1/17,$$

where the last equality follows from the same reasoning we used to bound $\Pr[A_r] = 1/17$. Now we have

$$\begin{aligned}
\Pr[A_r | \cap_{\ell \in J} A_\ell] &= \sum_{\vec{y}} \Pr[A_r | \cap_{\ell \in J} A_\ell, x_i = y_i \forall i \neq j] \Pr[x_i = y_i \forall i \neq j] \\
&= \sum_{\vec{y}} \frac{1}{17} \Pr[x_i = y_i \forall i \neq j] \\
&= 1/17 \\
&= \Pr[A_r].
\end{aligned}$$

where above the sum is over all values $y_i \in \{0, \ldots, 16\}$ for all $i \neq j$. This shows what we wanted to show. Hooray!

[a] Formally, we'd write

$$\Pr[A_r] = \sum_{y_i : i \neq j} \Pr[x_i = y_i \forall i \neq j] \cdot \Pr[A_r | x_i = y_i \forall i \neq j].$$

We just argued that each $\Pr[A_r | x_i = y_i \forall i \neq j] = 1/17$, so this says

$$\Pr[A_r] = \sum_{y_i : i \neq j} \Pr[x_i = y_i \forall i \neq j] \cdot 1/17 = 1/17.$$

4 More practice with derandomization via conditional expectation

I don't expect we will get to this today in class, but if you finish the rest early, try this part that we didn't get to last week!

Group Work

1. (Bonus) Let φ be a 3-CNF formula with n variables and m clauses, and 3 distinct variables in each clause. Use the method of derandomization via conditional expectation to give an efficient (polynomial in n, m) deterministic algorithm to find an assignment to φ so that at least a 7/8-fraction of the clauses are satisfied.
2. (Even more bonus) There is also a natural greedy algorithm for this problem:
   - For $i = 1, 2, \ldots, n$:
     - Assign x_i to be whichever value makes the most currently unsatisfied clauses true (breaking ties arbitrarily).

   In the previous example (maximizing the size of a cut), the algorithm we came up with was secretly the natural greedy algorithm. Is your algorithm from the previous part the same as this natural greedy algorithm? Is it better or worse?

Group Work: Solutions

1. We apply the method of deranomization by conditional expectation! We choose values for the variables $x_1, \ldots, x_n$ one at a time, and as before, provided that we have
$$\mathbb{E}[\text{number of sat clauses}|x_1, \ldots, x_{t-1}] \geq 7m/8$$
there is a choice for x_t so that
$$\mathbb{E}[\text{number of sat clauses}|x_1, \ldots, x_t] \geq 7m/8.$$
Thus, all our algorithm has to do is find it. Let X be the number of satisfied clauses. Then we can write
$$\mathbb{E}[X|x_1, \ldots, x_t] = \sum_C \Pr[C \text{ sat.}|x_1, \ldots, x_t].$$
Observe that for each clause C, we can actually compute
$$\Pr[C \text{ sat.}|x_1, \ldots, x_t]$$

in time $O(1)$: that's because there are at most 8 outcomes. Thus, we can compute the whole thing in time $O(m)$. So our algorithm is:

- For $t = 1, \ldots, n$:
  - Compute $\mathbb{E}[X|x_1, \ldots, x_{t-1}; x_t = T]$
  - If that is $\geq 7m/8$, set $x_t \leftarrow T$; otherwise set $x_t \leftarrow F$.
- Return $(x_1, \ldots, x_n)$.

The whole thing takes time $O(nm)$.

2. The "natural greedy algorithm" is worse than the algorithm from part (a). To see this, we can consider an example with $n = 8$ and $m = 8$. The algorithm from part (a) will come up with an assignment that satisfies 7 of the 8 clauses. The clauses are $(x_i \vee x_7 \vee x_8)$ for $i = 1, 2, \ldots, 6$, along with $(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_4} \vee \overline{x_5} \vee \overline{x_6})$. Each of $x_1, \ldots, x_6$ appears in exactly two clauses, once as-is and once negated, so the natural greedy algorithm can take an arbitrary choice for each of these variables. If we choose to set all of them to TRUE, then each of the last two clauses will be unsatisfied, so the greedy algorithm will satisfy only 6 out of the 8 clauses, worse than the algorithm from part (a). (If you don't like that this is based on breaking ties arbitrarily, consider repeating this example M times for some large M, and repeating its negation $M - 1$ times; as M grows, this will still approach satisfying only 3/4 of the clauses, instead of 7/8).