INFR11102: Computational Complexity 14/11/2019

Lecture 17: Approximate Counting

Lecturer: Heng Guo

1 Toda's theorem

One remarkable result, due to Seinosuke Toda [Tod91], shows that #P is more powerful than the whole of polynomial hierarchy.

Theorem 1. $\mathtt{PH} \subseteq \mathtt{P}^{\#\textsc{Sat}}$.

Toda's theorem is rather complicated and we will omit its proof. Details can be found in [AB09, Chapter 17.4].

On the other hand, given a Sat oracle, we can approximately count #Sat. Thus, approximately counting is at most NP-hard.

2 Isolation lemma

Our main tool for approximate counting is the isolation lemma, introduced by Valiant and Vazirani [VV86]. It also has a number of interesting applications related to BPP and the hardness of Sat when the formula has at most 1 satisfying assignment (called UniSat). It relies on a notion called *pairwise independent hash functions*.

2.1 Pairwise independent hash functions

Fix some (discrete) domain D and range R. Let $\mathcal{H} = \{H_i\}_{i\in[k]}$ be a family of functions, where for each $i \in [k]$, H_i is a function $D \to R$. The family $\mathcal{H}$ is said to be *pairwise independent* if for any two distinct $x, x' \in D$ and (not necessarily distinct) $y, y' \in R$,

$$\Pr_{i\in[k]}[H_i(x) = y \ \&\ H_i(x') = y'] = \frac{1}{|R|^2}. \tag{1}$$

Here the function H_i is chosen uniformly from the set $\mathcal{H}$. In other words, suppose $D = \{x_1, \cdots, x_\ell\}$. Consider the random variables $Y_j = H_i(x_j)$ for $j \in [\ell]$, where the randomness comes from choosing $i \in [k]$ uniformly at random. Then, Y_j is uniform over R, by summing over y' in (1). Moreover, any two Y_j and $Y_{j'}$ are independent (hence the name "pairwise independent"). It is not necessary that all $\{Y_j\}$ are mutually independent.

The requirement may look rather demanding, but next we show that there is a simple construction if D and R are both finite fields. Suppose $D = R = \mathbb{F}$. Let $\mathcal{H} = \{H_{a,b}\}_{a,b\in\mathbb{F}}$ and

$$H_{a,b}(x) = ax + b.$$

So the random function is chosen by picking two random element a and b in $\mathbb{F}$. For any distinct $x, x' \in \mathbb{F}$ and $y, y' \in \mathbb{F}$ as in the requirement above, the event in (1) happens if and only if

$$\begin{aligned} a &= (y - y')(x - x')^{-1}, \\ b &= y - (y - y')(x - x')^{-1} \cdot x. \end{aligned}$$

Since $x \neq x'$ are fixed, as well as y and y', the right hand side are two fixed elements in $\mathbb{F}$. So, the event happens with probability $\frac{1}{|\mathbb{F}|} \cdot \frac{1}{|\mathbb{F}|} = \frac{1}{|\mathbb{F}|^2}$ as required.

In the applications, the most common setup is to take $\mathbb{F} = \mathrm{GF}(2^n)$,[1] which is the unique finite field of size 2^n. We view its elements as $\{0,1\}$ vectors of length n, and we may take $D = R = \mathrm{GF}(2^n)$ in the construction above. Moreover, we can truncate the output up to size ℓ to get a pairwise independent hash function $\mathcal{H}$ from $\{0,1\}^n$ to $\{0,1\}^\ell$ for any $\ell \leq n$. This explains the name "hash function". We denote such a family as $\mathcal{H}_{n,\ell}$.

2.2 Isolation

Let f be a hash function $D \to R$. If $f(x) = f(y)$ for distinct $x, y \in D$, then this is a "collision". On the other hand, we say f *isolates* x (within a subset $S \subset D$) if there is no $y \in S$ such that $f(y) = f(x)$. Moreover, we say that a set F of functions *isolates* x if $\exists f \in F$ isolates x, and F isolates a set $S \subseteq D$ if for any $x \in S$, F isolates x.

The isolation lemma says that, given a pairwise independent hash family and $S \subseteq D$, if S is "small" (in some precise ways), then with positive probability a random hash function isolates S. Otherwise S is too big and isolate won't happen.

Lemma 2. *Let $S \subseteq \{0,1\}^n$, $\mathcal{H}_{n,k}$ be a pairwise independent hash family from $\{0,1\}^n$ to $\{0,1\}^k$, and $m \geq k$. Choose $H_1, \cdots, H_m$ independently and uniformly at random from $\mathcal{H}_{n,k}$. We have the following:*

1. *if $|S| \leq 2^{k-1}$, then*

$$\Pr\left[\{H_1, \ldots, H_m\} \text{ isolates } S\right] \geq 1 - \frac{1}{2^{m-k+1}};$$

2. *if $|S| > m2^k$, then*

$$\Pr\left[\{H_1, \ldots, H_m\} \text{ isolates } S\right] = 0.$$

Proof. Item (2) follows easily from a counting argument. For each H_i, it can isolate at most $2^k - 1$ elements. (We map $2^k - 1$ elements to distinct locations, and all others to the last spot in $\{0,1\}^k$.) Hence, $\{H_1, \ldots, H_m\}$ isolates at most $m(2^k - 1) < m2^k = |S|$ elements.

[1] GF stands for "Galois field", which is just a synonym of "finite field". It is known that a finite field exists if and only if its size is p^n for some prime p and natural number $n \geq 1$. Moreover, such a field is unique. The simplest example is $\mathrm{GF}(2) = \mathbb{F}_2$ which contains only $\{0,1\}$.

For Item (1), by the definition of pairwise independent hash families,

$$\Pr_{H\in\mathcal{H}_{n,k}}[H(x)=H(y)] = \sum_{z\in\{0,1\}^k} \Pr_{H\in\mathcal{H}_{n,k}}[H(x)=H(y)=z] = 2^{-k}.$$

Hence, by a union bound,

$$\Pr_{H\in\mathcal{H}_{n,k}}[\ H \text{ does not isolate } x\ ] \le \sum_{y\in S\setminus\{x\}} \Pr_{H\in\mathcal{H}_{n,k}}[H(x)=H(y)] \le \frac{|S|}{2^k} \le 1/2.$$

Since $H_1,\cdots,H_m$ are chosen independently,

$$\Pr_{H_1,\ldots,H_m\in\mathcal{H}_{n,k}}[\ \text{none of } \{H_i\} \text{ isolates } x\ ] \le 2^{-m}.$$

Once again, by a union bound,

$$\begin{aligned}
&\Pr_{H_1,\ldots,H_m\in\mathcal{H}_{n,k}}[\{H_1,\ldots,H_m\} \text{ does not isolate } S] \\
\le &\sum_{x\in S} \Pr_{H_1,\ldots,H_m\in\mathcal{H}_{n,k}}[\{H_1,\ldots,H_m\} \text{ does not isolate } x] \\
\le &\ |S|\,2^{-m} \le 2^{-(m-k+1)}.
\end{aligned}$$

Taking the complement, we get Item (1). □

3 Approximate counting

An interesting application of the isolation lemma is that given a SAT oracle, we can approximately count #SAT. This is firstly shown by Sipser [Sip83] and is simplified by Valiant and Vazirani [VV86] using the isolation lemma. Note that we may replace the SAT oracle by an arbitrary NP-complete oracle and #SAT by any problem in #P, but we will stick to SAT and #SAT for clarity.

Let S be the set of satisfying assignments to a formula φ with n variables, and $Z := |S|$. Then $0 \le Z \le 2^n$. We may assume that $Z \neq 0$, since this can be verified by a simple oracle call. The basic idea is to find an integer $0 \le k \le n$ such that $Z \approx 2^k$. There must exist an integer k_S such that $2^{k_S-1} \le Z \le 2^{k_S}$. Finding this k_S yields a good approximation to Z (which can be amplified later).

We will test all $k \in [n+1]$, and set $m = 2n$ in Lemma 2. Namely, we randomly pick $2n$ hash functions $H_1,\cdots,H_{2n} : \{0,1\}^n \to \{0,1\}^k$. By Lemma 2, if $k \ge k_S + 1$, isolation happens with probability at least $1-2^{-n}$. We use the SAT oracle to verify whether isolation happens. In other words, for each k, we ask whether "$\forall x \in S$, one of H_i isolates x?" This can be expressed as a logical formula as follows. Firstly,

$$H_i \text{ isolates } x \Leftrightarrow \forall y \text{ s.t. } \varphi(y)=1, \text{ and } H_i(x) \neq H_i(y).$$

Then we can express the query as

$$\forall x\ \exists i\ \forall y \text{ if } \varphi(x) = \varphi(y) = 1 \text{ and } x \neq y, \text{ then } H_i(x) \neq H_i(y).$$

Although this looks like an $\forall\exists\forall$-expression, there are only $2n$ choices of i, and can thus be rewritten with only one layer of $\forall$ quantifiers and a polynomial blow-up in its size. We take its complement to get a proper SAT query.

By going through all of k, we found the smallest one such that isolation happens. Denote it by k_0. The probability that isolation does not happen for all $k \leq n+1$ is exponentially small by Item (1) of Lemma 2. In particular, isolation happens for $k = k_S+1$ with probability at least $1 - 2^{-n}$. Thus $k_0 \leq k_S + 1$ with probability at least $1 - 2^{-n}$.

On the other hand, by Item (2) of Lemma 2, we have that $Z \leq 2n2^{k_0}$. Let $\widehat{Z} := 2n2^{k_0}$. We have that with probability at least $1 - 2^{-n}$,

$$\widehat{Z} \geq Z \geq 2^{k_S - 1} \geq 2^{k_0 - 2} = \frac{\widehat{Z}}{8n}.$$

Hence, this gives us a randomized algorithm approximate Z within a ratio of $O(n)$.

We can do even better by an amplification trick. Given φ, we construct a formula

$$\varphi^{(t)} := \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_t,$$

where each φ_i for $i \in [t]$ is an independent copy of φ, and the integer t will be set later. Notice that in order to obtain independent copies, each variable in φ is duplicated t times in $\varphi^{(t)}$, namely, each x_i is replaced by $x_i^{(1)}, \cdots, x_i^{(t)}$. Denote by $Z^{(t)}$ the number of solutions to $\varphi^{(t)}$. It is easy to see that

$$Z^{(t)} = Z^t.$$

We use the randomized algorithm above to approximate $Z^{(t)}$, getting an estimate $\widehat{Z^{(t)}}$. Since $\varphi^{(t)}$ contains tn variables,

$$\frac{1}{8nt} \leq \frac{Z^t}{\widehat{Z^{(t)}}} \leq 1$$

Let $\widehat{Z} = \left(\widehat{Z^{(t)}}\right)^{1/t}$ be our final estimate. Then,

$$\frac{1}{(8nt)^{1/t}} \leq \frac{Z}{\widehat{Z}} \leq 1$$

The function $(8nt)^{1/t}$ goes to 1 very quickly as t goes to infinity. Suppose we want to estimate Z within $(1 \pm \varepsilon)$ precision, then all we need is

$$\frac{1}{(8nt)^{1/t}} \geq \frac{1}{1+\varepsilon} \geq 1 - \varepsilon,$$

or equivalently,

$$(1+\varepsilon)^t \geq 8nt.$$

This can be achieved by letting $t = O\left(\frac{n}{\varepsilon}\right)$ as the left is polynomial in n, and the right is exponential in n. In summary, we have the following theorem.

Theorem 3. *There is a randomized algorithm with a* Sat*-oracle, that approximates* #Sat *within* $1 \pm \varepsilon$ *and runs in time polynomial in* n *and* ε^{-1}*.*

Such an algorithm is usually called a *fully polynomial-time randomized approximation scheme* (FPRAS).

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335. ACM, 1983.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.