6.842 Randomness and Computation October 1, 2017

Lecture 11

Lecturer: Ronitt Rubinfeld *Scribe: Di-Chia Chueh*

1 Today's topic

- Reducing randomness via random walks on very nice graphs
- Analyzing random bits usage and upper-bounding error

2 Reduce error in randomized algorithm

Definition 1 *For a decision problem L: given a* randomized algorithm A with one-sided error *such that*

1. $\forall x \in L$, $Pr_\$[A(x) = 1] \geq 0.99$
2. $\forall x \notin L$, $Pr_\$[A(x) = 0] = 1$

where \$ denotes the coin toss, $A(x)$ is the output when taking x as input (of size n), and assume we need $r \equiv r(n)$ random bits for algorithm A.

Goal Note that algorithm A has one-sided error only when input x is in the language, with error probability of an arbitrary fixed constant. Without loss of generosity here we pick the error to be 0.99 for the rest of our notes. Our goal is to get an error of 2^{-k} for some constant k. Possible methods include:

1. n naive way is to run algorithm A repeatedly for k times, output "$x \in L$" if ever seen A accepts (stating $x \in L$), or else output "$x \notin L$". Note that it requires $O(k \cdot r)$ bits, where r denotes number of bits A uses.
2. Previously *(in lecture 4,5)* we used pair-wise independence random strings, which requires $O(k + r)$ random bit.
3. We can leverage random walks to choose random bits to use.

Theorem 2 *It requires only $O(k) + r$ random bits in random walk method with k steps. Thus in some case it's better than pair-wise random bits method in terms of random bits usage.*

Method	Number of random bits used
Initialize new random bits k times	$O(k \cdot r)$
Use pair-wise independence	$O(k + r)$
Use random walk	$O(k) + r$

Table 1: Comparison of required random bits

3 Reduce randomness via random walk

3.1 Algorithm and random bits analysis

Plan Our plan is to create a graph G (which is built to help select the random strings; is independent of A), with every possible random bit string $m \in \{0,1\}^m$ associate with a node in G. Then picking several random strings is now equivalent to picking several nodes in G. We can do random walk on G and decide which random string to pick in each step.
All we need is to pick several strings with at least one of which is a good random string, such that algorithm A will accept with that string when input is in L; which now correspond to selecting several nodes with one of which is a good node, that is, the code is associated with a good string.

Definition 3 *Define graph $G = (V, E)$ such that $|V| = 2^r$, and such that G is d-regular for some constant d, and that (undirected) G is connected, non-bipartite. And we have transition matrix P for* random walk *on G such that P has a set of eigenvalues satisfying $\lambda_2 \leq \frac{1}{10}$. We label each node a binary representation of number in $[0, 2^r - 1]$ corresponding to each random string: $s(v) \in \{0,1\}^r$, $\forall v \in V$.*

Observation 4 *Now that G exists, we can find random neighbor in G in reasonable time and $O(\log d) = O(1)$ random bits.*

The algorithm With input x, we run the new algorithm A' as follows:

	Random bits used:
1. Pick start node w in G, $s(w) \in \{0,1\}^r$.	r
2. Repeat k times: $w \leftarrow$random neighbor of w; run $A(x)$ with random string $s(w)$; if A accepts (says "$x \in L$"), than outputs "$x \in L$".	$k \cdot O(1)$
3. Otherwise output "$x \notin L$".	Total: $r + O(k)$

3.2 Error analysis

Claim 5 *Error of new algorithm $A' \leq \left(\frac{1}{5}\right)^k$ for $x \in L$, while still has zero error for $x \notin L$ (which is obvious).*

Proof Idea We separate graph G into two regions, one with good nodes with good string that will that $A(x)$ answer correctly if x is in L; and the other with bad nodes. The algorithm fails to answer correctly if the random walk are always within the bad region (which is unlikely). ■

Definition 6 *Let $B \equiv \{w | A(x)$ with random bits w is incorrect$\}$ be the set of random strings in* bad region*, then define $|V| \times |V|$ diagonal (indicator) matrix N with $\forall w \in V$, the $s(w)$-th diagonal element*

$$N_w = \begin{cases} 1, \text{ if } w \in B \\ 0 \text{ , otherwise} \end{cases} .$$

Proof Given any ($|V| \times 1$ vector) probability distribution q, we'd like to filter out the probability mass on good locations and only retain the bad ones, i.e. qN. When taking first-norm of the filtered vector, $\|qN\|_1$ = sum of the bad probability = total probability at a bad code.
We can see that $\|qPN\|_1$ = probability starting at distribution q, take one step, and landing at another bad node; $\|qPNPN\|_1$ = then taking another step and still land at a bad code; thus $\|q(PN)^k\|_1$ = probability of never see good codes in k steps.

Lemma 7 $\forall$ *vector π, $\|\pi PN\|_2 \leq \frac{1}{5}\|\pi\|_2$. Note that it's true for arbitrary π, not necessary need to be stationary distribution.*

Fact 8 $\forall$ *vector p, $\|p\|_1 \leq \sqrt{p\text{'s dimension size}}\,\|p\|_2$.*

Thus when we start with the uniform probability distribution p_0, using previous lemma and fact, we have

$$Pr[\text{error}] = \|p_0(PN)^k\|_1 \leq \sqrt{2^r}\|p_0(PN)^k\|_2 \leq \frac{\sqrt{2^r}}{5}\|p_0(PN)^{k-1}\|_2$$

$$\leq \frac{\sqrt{2^r}}{5^k}\|p_0\|_2 = \frac{\sqrt{2^r}}{5^k}\sqrt{\sum\left(\frac{1}{2^r}\right)^2} = \frac{\sqrt{2^r}}{5^k} \times \frac{1}{\sqrt{2^r}} = \frac{1}{5^k}$$

, which means that using algorithm A' we can reduce the one-sided error to 5^{-k}, while using $O(k) + r$ random bits. ■

3.3 Finishing the Proof to Upper-bound the Error

Now the only thing left is to proof the inequality in the lemma, the decrease of bad code probability that after one iteration of transition.

Linear Algebra Revisited First we set (orthonormal basis) $v_1,\cdots,v_{2^r}$ to be eigenvectors of P corresponding to eigenvalues (in decreasing order) $\lambda_1,\cdots,\lambda_{2^r}$, such that $\|v_1\|_2 = 1$, and we have that $v_1 = \frac{1}{\sqrt{2^r}}\overrightarrow{1}$ (where $\overrightarrow{1}$ is the all-one vector). We assume $\pi = \sum_{i=1}^{2^r} \alpha_i v_i$. We note the following two facts:

$$\|\pi\|_2 = \sqrt{\sum_i \alpha_i^2} \geq |\alpha_j|, \forall j \tag{1}$$

$$\forall w \text{ with vector elements } w_i, \|wN\|_2 = \sqrt{\sum_{i \in B} w_i^2} \leq \sqrt{\sum_i w_i^2} = \|w\|_2 \tag{2}$$

Proof of Lemma 7 Now we have

$$\|\pi PN\|_2 = \|\left(\sum_{i=1}^{2^r} \alpha_i v_i\right) PN\|_2 = \|\sum_{i=1}^{2^r} \alpha_i \lambda_i v_i N\|_2 \le \overbrace{\|\alpha_1 \lambda_1 v_1 N\|_2}^{\text{part A}} + \overbrace{\|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N\|_2}^{\text{part B}} \tag{3}$$

The inequality is due to triangular inequality. For part A,

$$\|\alpha_1 \lambda_1 v_1 N\|_2 = \|\alpha_1 v_1 N\|_2 = |\alpha_1| \|v_1 N\|_2 = |\alpha_1| \sqrt{\sum_{i \in B} \left(\frac{1}{\sqrt{2^r}}\right)} = |\alpha_1| \sqrt{\frac{|B|}{\sqrt{2^r}}} = \frac{|\alpha_1|}{10} \le \frac{\|\pi\|_2}{10} \tag{4}$$

where the first equality holds since $\lambda_1 = 1$, the third equality holds due to equation (2) and the value of v_1, the fourth equality holds because we pick the one-sided error to be 0.99; whereas the inequality holds from equation (1). For part B,

$$\|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N\|_2 \le \|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i\|_2 = \sqrt{\sum_{i=2}^{2^r} (\alpha_i \lambda_i)^2} \le \sqrt{\frac{\sum_i \alpha_i^2}{100}} = \frac{1}{10}\|\pi\|_2 \tag{5}$$

where the second inequality holds because eigenvalues $\lambda_i \le \frac{1}{10}$ with $\forall i \ge 2$; whereas the last equality holds due to equation (1). Combining (4) and (5) with (3), we have $\|\pi PN\|_2 \le \frac{\|\pi\|_2}{10} + \frac{\|\pi\|_2}{10} = \frac{\|\pi\|_2}{5}$, and this complete the proof of the lemma. ■

Putting everything together, we have shown a method to reduce randomness via random walks on very nice graphs.