6.895 Randomness and Computation April 9, 2008

Lecture 17

Lecturer: Ronitt Rubinfeld *Scribe: Bill Jacobs*

1 Derandomization of algorithms with one-sided error

This lecture describes a means of reducing the number of random bits required to achieve exponentially low error using a randomized algorithm with a one-sided error. Specifically, we are interested in algorithms A intended to determine membership in a language L, for which:

- If $x \in L$, then $Pr(A(x) \text{ says } x \notin L) \leq \frac{1}{100}$.
- If $x \notin L$, then $Pr(A(x) \text{ says } x \in L) = 0$.

Say that A uses at most r random bits for all inputs of size at most n. To get a (one-sided) error of at most 2^{-k}, we may take a few approaches:

Approach	**Number of random bits required**
Run A $O(k)$ times	$O(kr)$
Run A $O(k)$ times using pairwise-independent random bits	$O(k+r)$
Use random walks on graphs to choose random bits (the technique described in this lecture)	$r + O(k)$

2 Using random walks to reduce randomness

For the algorithm in this lecture, we use a d-regular (for a constant d), connected non-bipartite graph G with 2^r nodes numbered from 0 to $2^r - 1$. We order the eigenvalues λ of the transition matrix P for a random walk on G so that $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|$. Recall from last lecture that $\lambda_1 \leq 1$, a claim we stated without proving. Also, we know from last lecture that $|\lambda_2| \leq \frac{1}{10}$, since G is d-regular and $\bar{\Pi}$ is uniform.

Our algorithm operates as follows on an input x:

1. Pick a random starting node w in G.
2. Run $A(x)$ using the bits in w for the random bits.
3. Do k times:
   (a) Take a random step according to the transition matrix P.
   (b) Set w to be the number of the current node.
   (c) Run $A(x)$ using the bits in w for the random bits.
4. If A ever outputs $x \in L$, output $x \in L$; otherwise, output $x \notin L$.

3 Proof of the algorithm's correctness

3.1 Initial observation

For a fixed $x \in L$, let B be $\{w|A(x)$ outputs "$x \notin L$" using the bits in w for the random bits$\}$. This is the "bad set" of random bits. Our algorithm will fail precisely when the random walk stays entirely in the set B. Because A has one-sided error, we know that $|B| \leq \frac{2^r}{100}$.

Let N be the diagonal matrix defined so that along the main diagonal, $N_{w,w} = \begin{cases} 1, & w \in B \\ 0, & \text{otherwise} \end{cases}$. Note that N will look something like this:

$$\begin{bmatrix} 1 & & & & & & \\ & 0 & & & & & \\ & & 1 & & & & \\ & & & 0 & & & \\ & & & & 0 & & \\ & & & & & \ddots & \end{bmatrix}$$

If we have a distribution p of nodes in G, the product pN zeroes out any of the "good nodes", that is, the indices w that are witnesses to x's membership in L. More precisely, $(pN)_w = \begin{cases} p_w, & x \in B \\ 0, & \text{otherwise} \end{cases}$. It follows that $|pN|_1$ (also denoted $|pN|$) is equal to $Pr_{w \in p}(w \in B)$ (that is, $|pN|$ is the probability that a value w selected according to the distribution p is in B). Also, observe that

$$|pNPN| = Pr_{w \in p}(w \in B \text{ and a node that is one random step from } w \text{ is in } B)$$

Continuing the pattern,

$$|pN\underbrace{PNPN...PN}_{k}| = Pr_{w \in p}(w \in B \text{ and each node in a random walk of } k \text{ steps from } w \text{ is in } B)$$

This observation serves as a starting point for our proof of correctness.

3.2 Lemma

Our proof makes use of the following lemma: $\forall \Pi, ||\Pi PN||_2 \leq \frac{1}{5}||\Pi||_2$. (Note that $||X||_2$ is also denoted $||X||$.) From this, it easily follows that $\forall \Pi, k \geq 0, ||\Pi(PN)^k|| \leq \left(\frac{1}{5}\right)^k ||\Pi||$.

3.3 How the lemma proves the algorithm's correctness

Let p_0 be the uniform distribution over all nodes, the initial distribution in the random walk. We have that $Pr(\text{all } k \text{ calls to } A \text{ are bad}) = |p_0 N\underbrace{PNPN...PN}_{k})| \leq |p_0(PN)^k|$. Using the fact that the L_1 norm of an n-dimensional vector is at most $\sqrt{n}$ times the L_2 norm of the vector, we continue with $|p_0(PN)^k| \leq \sqrt{2^r}||p_0(PN)^k|| \leq \sqrt{2^r}||p_0||\left(\frac{1}{5}\right)^k$. In addition, we know that $||p_0|| = \sqrt{\sum_{i=1}^{2^r}\left(\frac{1}{2^r}\right)^2} = \sqrt{\frac{1}{2^r}}$, so we may conclude by observing $\sqrt{2^r}||p_0||\left(\frac{1}{5}\right)^k \leq \left(\frac{1}{5}\right)^k \leq 2^{-k}$.

3.4 Proof of the lemma

Let v_i are eigenvectors of P with magnitude 1 with associated eigenvalues λ_i so that $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_{2^r}|$. Take some distribution Π. Recall from last lecture that eigenvectors of a transition matrix form an orthonormal basis. So we can write Π as $\sum_{i=1}^{2^r} \alpha_i v_i$. We have that $||\Pi PN|| = \left|\left|\sum_{i=1}^{2^r} \alpha_i v_i PN\right|\right| = \left|\left|\sum_{i=1}^{2^r} \alpha_i \lambda_i v_i N\right|\right|$. Using Cauchy-Schwartz, we further have that $\left|\left|\sum_{i=1}^{2^r} \alpha_i \lambda_i v_i N\right|\right| \leq ||\alpha_1 \lambda_1 v_1 N|| + \left|\left|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N\right|\right|$. Call the expression $||\alpha_1 \lambda_1 v_1 N||$ (1) and the expression $\left|\left|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N\right|\right|$ (2). We will show that both (1) and (2) are at most $\frac{1}{10}||\Pi||$, from which it follows that their sum is at most $\frac{1}{5}||\Pi||$, proving the lemma.

We know that $|\lambda_1| = 1$ and $v_1 = \left(\frac{1}{\sqrt{2^r}}, \frac{1}{\sqrt{2^r}}, ..., \frac{1}{\sqrt{2^r}}\right)$. So we may evaluate expression (1) as follows:

$$\begin{aligned} ||\alpha_1 \lambda_1 v_1 N|| &= ||\alpha_1 v_1 N|| = |\alpha_1| \cdot ||v_1 N|| = |\alpha_1| \sqrt{\sum_{i \in B} \left(\frac{1}{\sqrt{2^r}}\right)^2} \\ &= |\alpha_1| \sqrt{\frac{|B|}{2^r}} \leq \frac{|\alpha_1|}{10} \leq \frac{1}{10}||\Pi||. \end{aligned}$$

For expression (2), using the fact that $\frac{1}{10} \geq |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_{2^r}|$, we have

$$\left|\left|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N\right|\right| \leq$$

$$\left|\left|\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i\right|\right| \leq$$

$$\sqrt{\sum_{1=1}^{2^r} (\alpha_i \lambda_i)^2} \leq$$

$$\sqrt{\frac{1}{100} \sum_{i=1}^{2^r} \alpha_i^2} \leq$$

$$\frac{1}{10}||\Pi||$$

4 Derandomizing S-T connectivity (Reingold)

Recall from last lecture that in the problem of S-T connectivity, we want to determine whether two nodes S and T of an undirected graph G are connected. In this lecture, we consider a logarithmic-space algorithm for computing S-T connectivity in an easy case.

We will consider (N, D, λ)-graphs G. A (N, D, λ)-graph has

- N nodes
- Constant degree D
- Second-largest absolute value of an eigenvalue $|\lambda_2|$ of the transition matrix at most λ

To construct our algorithm, we use two well-known facts. Our first well-known fact (Tanner, Alan-Milman) stipulates

$$\forall \lambda < 1, \exists \varepsilon > 0 \text{ such that } \forall (N, D, \lambda)\text{-graphs } G, \forall S \text{ with } |S| \leq \frac{N}{2}, |\Gamma(S)| \geq (1+\varepsilon)|S|.$$

In other words, (N, D, λ)-graphs are expanders. (Here, $\Gamma(S)$ is defined to be the union of all nodes in S and nodes adjacent to a node in S.)

A second well known fact is a corollary of this first fact. It holds that (N, D, λ)-graphs have a diameter of at most $O(\log N)$. We prove this by considering two nodes S and T from a (N, D, λ)-graph G. Let ε be some constant implied by the preceding fact. Say we start with the node S, then add the set of nodes adjacent to S, then add the set of nodes adjacent to one of these nodes, and so on. At each iteration of this procedure, we have at least $1 + \varepsilon$ times as many nodes as in the previous iteration. Thus, after at most $\log_{(1+\varepsilon)} \frac{N}{2}$ iterations, we will have reached at least $\frac{N}{2}$ nodes. Similarly, by taking at most $\log_{(1+\varepsilon)} \frac{N}{2}$ steps from T, we can reach at least $\frac{N}{2}$ nodes. So there must be some node within $\log_{(1+\varepsilon)} \frac{N}{2} = O(\log N)$ steps of both S and T. (Actually, this is not completely correct, as they could be two disjoint sets of $\frac{N}{2}$ nodes. This can be fixed without much difficulty.)

Using this, the following logarithmic-space algorithm will determine S-T connectivity in a (N, D, λ)-graph: enumerate all paths of length at most $l = O(\log N)$ starting at S, and if a path ever reaches T, output "connected"; otherwise, output "not connected". The algorithm uses $O(l \log D) = O(\log D \log N)$ space.

This algorithm is only useful for an easy type of graph, but it will be built on in the next lecture.