CS 710: Complexity Theory 11/1/2011

Lecture 16: Error Reduction

Instructor: Dieter van Melkebeek Scribe: Aaron Gorenstein

Last lecture we introduced expanders, graphs with few edges that act like graphs with many edges. We discussed algebraic and combinatorial definitions, and introduced how expanders are used to conserve a resource in BPP computation: the number of random bits used in error reduction.

This lecture we review the definitions and establish their equivalence. We construct a class of expander graphs, and apply those to implement two forms of error reduction. One form is entirely deterministic—after the first run uses r random bits, it does not use any more but can still reduce the error probability. Another uses a few random bits per additional run, and greatly reduces the error probability.

1 Expander review: two definitions and equivalence

Combinatorial Definition $G = (V, E)$ is a (k, c)-expander if $\forall B \subseteq V, |B| \leq k$, each such B is also c-expanding, meaning $|\Gamma(B)| \geq c\,|B|$. Recall that $\Gamma(B)$ is the set of neighbors of the vertices in B.

Algebraic Definition If we represent the d-regular graph G with the normalized adjacency matrix A, we define the 2^{nd} largest eigenvalue as $\lambda(G) = \text{MAX}(\{|\lambda| \mid \exists \text{ an eigenvector } v \text{ of } A, Av = \lambda v \wedge v \perp u\})$.

Linking the two definitions Recall we define $|V| = N$. Consider the two theorems, which we will refrain from proving:

Theorem 1. *If G is an $(\frac{N}{2}, c)$-expander, then $\lambda(G) \leq f(c, d)$, where $f(c, d) < 1$ for $c > 1$.*

Theorem 2. *If $\lambda(G) \leq \lambda$, then G is an $(\frac{N}{2}, c)$-expander where $c \geq g(\lambda)$ and $g(\lambda) > 1$ if $\lambda < 1$.*

Consider: then for a family of d-degree graphs, we have the following statement:

$$\exists c_0 > 1 \text{ s.t. } c \geq c_0 \Leftrightarrow \exists \lambda_0 < 1 \text{ s.t. } \lambda \leq \lambda_0$$

In other words, we can bound away any c above 1 if and only if we can bound any λ below 1, for c relating to the combinatorial definition, and λ the algebraic.

2 Explicit Construction

For all their desirable properties, do expanders actually exist? If we randomly make a graph of fixed degree d it is usually an expander. But that thoroughly defeats our purpose: *conserving* random bits.

We would like an *explicit, deterministic* construction of expanders. Particularly, given an index of a vertex, we would like to compute the neighbors of that vertex in poly-log(n) time. Here is an example based off of pairs of integers modulo m.

Example: $V = \mathbb{Z}_m \times \mathbb{Z}_m$, so a single vertex $v = (x, y) \mod m$.

Each vertex has exactly 8 neighbors:

$$\begin{array}{ll} (x \pm y, y) & (x \pm (y+1), y) \\ (x, y \pm x) & (x, y \pm (x+1)) \end{array}$$

⊠

3 Application to Error Reduction

We consider two applications of expanders to error reduction. The first is **deterministic error reduction**, in which we do not use any more random bits than the original r, but repeat our BPP computation again with deterministically-computed neighbors of r, and we prove that we can still reduce the error from its initial ϵ_0. The general strategy is to view all random bit sequences r as vertices of an expander graph. Given the vertex r, we compute its neighbors (deterministically) and try those. The majority vote determines if we accept or reject.

The second is **randomness-efficient error reduction**. It is quite similar to the deterministic method, but instead of choosing all neighbors, we take a short *random walk* from our starting point r. For a walk of length t, this requires only $O(t)$ more random bits, because we have a fixed-degree graph. This is how we can easily get to error 2^{-k} using only $r + O(k)$ random bits. Again, we decide based on the majority vote of those polled.

Before we move on, let us consider two properties of these expanders.

1. $\lambda(G)$ (the second-largest eigenvalue) dictates how quickly it converges to the uniform distribution. Recall from the previous lecture: $\|A^t p - u\|_1 \leq \sqrt{N}\lambda^t$. This formula expresses that idea.

2. There is also the *expander mixing lemma.* The term "mixing" relates to Markov chains.

Lemma 1 (Expander Mixing Lemma)**.** *For every pair of subsets $S, T \subseteq V$,*

$$\left| \frac{|E(S,T)|}{dN} - \mu(S)\mu(T) \right| \leq \lambda \sqrt{\mu(S)(1-\mu(S))\mu(T)(1-\mu(T))}. \tag{1}$$

The term $E(S, T)$ is the number of edges between the vertices subsets S, T. Recall that $\mu(S) = \frac{|S|}{|V|}$. In a sense, this lemma establishes a measure for how "good" our expander is. If the right-hand value is small (so the difference is small), that means choosing just *one* random vertex and a random neighbor is "almost as good as" choosing *two* random vertices. Recall that with a constant degree, the number of random bits to choose a random neighbor is much less than to choose an entirely new vertex. In other words, $2 \log |V| > \log |V| + \log d$.

Now we must establish such a bounds.

Proof. Expander Mixing Lemma

We must translate $|E(S, T)|$ into algebraic terms. Let $A(G)$ be the normalized adjacency matrix of G. Recall that $A(G)$ is symmetric and real implying that it has a full orthonormal eigenbasis. The number of edges between two sets S and T can be written in terms of their relative characteristic vectors (i.e. $\chi_{S,i} = 1$ iff $v_i \in S$ and $\chi_{S,i} = 0$ otherwise):

$$|E(S,T)| = \chi_S^T (dA) \chi_T \tag{2}$$

By definition of A, dA is the standard adjacency matrix for G. Both χ_S and χ_T can be rewritten in terms of their components parallel and perpendicular to the uniform vector u. Recall $u = (\frac{1}{N}, \frac{1}{N}, ..., \frac{1}{N})$ is an eigenvector corresponding to the eigenvalue 1.

$$\begin{aligned}
\chi_T &= \chi_T^{\parallel} + \chi_T^{\perp} \\
(\chi_T, u) &= (\chi_T^{\parallel}, u) + (\chi_T^{\perp}, u) \\
\chi_T^{\perp} \perp u &\rightarrow (\chi_T^{\perp}, u) = 0 \\
\chi_T^{\parallel} \parallel u &\rightarrow \chi_T^{\parallel} = \alpha \cdot u \\
(\chi_T^{\parallel}, u) &= \alpha(u, u) \\
\frac{|T|}{N} &= \alpha \frac{1}{N} \\
\alpha &= |T| \\
\text{So:} & \\
\chi_T &= \chi_T^{\parallel} + \chi_T^{\perp} \\
\chi_T &= |T| \cdot u + \chi_T^{\perp}
\end{aligned}$$

Now we have $A\chi_T = |T|\, u + A\chi_T^{\perp}$. If we repeat this process for χ_S we can re-write our formula for $|E(S,T)|$:

$$\begin{aligned}
|E(S,T)| &= (\chi_S^{\parallel} + \chi_S^{\perp})(dA)(\chi_T^{\parallel} + \chi_T^{\perp}) \\
&= \chi_S^{\parallel} dA \chi_T^{\parallel} + \chi_S^{\perp} dA \chi_T^{\perp} \\
&= d\chi_S^{\parallel}\chi_T^{\parallel} + \chi_S^{\perp} dA \chi_T^{\perp} \\
&= d\frac{|S|\,|T|}{N} + \chi_S^{\perp} dA \chi_T^{\perp}.
\end{aligned} \tag{3}$$

The second and third lines follow from the first because $\chi^{\parallel}$ is an eigenvector of A causing the first term to simplify. The cross terms vanish because, for example, $A\chi_T^{\perp}$ creates a vector that's orthogonal to u, and so the inner product is 0. Dividing by dN, moving terms around and taking the absolute value gives:

$$\begin{aligned}
\left|\frac{|E(S,T)|}{dN} - \mu(S)\mu(T)\right| &= \left|\frac{\chi_S^{\perp}(dA)\chi_T^{\perp}}{dN}\right| \\
&\leq \frac{\|\chi_S^{\perp}\|_2 \cdot \lambda \cdot \|\chi_T^{\perp}\|_2}{N}
\end{aligned} \tag{4}$$

The second line is reached by applying Cauchy-Schwarz to the RHS and using the fact that A decreases the magnitude of $\chi_T^{\perp}$ by at least λ as there is no component of $\chi_T^{\perp}$ along u. Applying the Pythagorean theorem and some simple algebra to $\|\chi_S\|_2$ we can derive the value of $\|\chi_S^{\perp}\|_2$:

$$\begin{aligned}
\|\chi_S\|_2^2 &= \|\chi_S^{\parallel}\|_2^2 + \|\chi_S^{\perp}\|_2^2, \text{ therefore} \\
|S| &= |S|^2\frac{1}{N} + \|\chi_S^{\perp}\|_2^2, \text{ and} \\
\|\chi_S^{\perp}\|_2^2 &= |S|(1 - \mu(S)), \\
\|\chi_S^{\perp}\|_2 &= \sqrt{|S|(1 - \mu(S))}.
\end{aligned} \tag{5}$$

Substituting this back in for $\|\chi_S^\perp\|_2$ and $\|\chi_T^\perp\|_2$ and pulling the factor of N into the square root completes the proof. □

4 Deterministic Error Reduction

Theorem 3 (Deterministic Error Reduction)**.** *Given a* BPP *algorithm that uses r random bits on input x we can reduce the error to $\leq \epsilon$ using* poly$(\frac{1}{\epsilon})$ *runs of the original algorithm and r random bits.*

Recall our new procedure: run M on all d neighbors of our random bit string of length r. Consider the set of failure nodes who have over half of their neighbors also as failure nodes. As we consider the majority vote for our new decision, those are exactly our "failure" nodes. The number of such nodes over the total size of the graph is our failure rate, our new ϵ.

Proof. Let B denote the set of all "bad" random strings. Formally:

$$B = \{\rho \in \{0,1\}^r | M(x,\rho) \text{ Errs on input } x\} \tag{6}$$

Note that the inner computation is deterministic—ρ represents our random bits. The probability of our method failing is:

$$B'(x) = \{\rho \in \{0,1\}^r | \text{ over } \frac{d}{2} \text{ of } N(\rho) \text{ errs on } x\} \tag{7}$$

In other words, we will fail only if over half of the neighbors of ρ are "bad."

As our algorithm is in BPP, we know that $\mu(B) \leq \epsilon_0$ (the fraction of the bad strings is directly proportional to the original error). If our goal is to reduce the error to less than ϵ, then $\mu(B') \leq \epsilon$.

Our error is the fraction of times a bad ρ has over half of its neighbors in B'. We will then use the expander mixing lemma to place bounds on that value. As we're considering the times when we're wrong, at least half of B''s neighbors are bad, hence $|E(B',B)| \geq |B'|\frac{d}{2}$. We re-express our definition:

$$\frac{|E(B',B)|}{dN} \geq \frac{|B'|\frac{d}{2}}{dN} = \frac{\mu(B')}{2} \tag{8}$$

This serves to simplify the left-hand-side of the expander mixing lemma application. And if instead of considering ρ's deterministically-chosen neighbors, we chose a new random variable:

$$\mu(B')\mu(B) \leq \mu(B')\epsilon_0 \leq \mu(B')/2 \tag{9}$$

Now we can compare these two options with the expander mixing lemma, noting that we do not need absolute values because of the last relation in 9.

$$\frac{\mu(B')}{2} - \mu(B')\epsilon_0 = \mu(B')\left(\frac{1}{2} - \epsilon\right) \leq \lambda\sqrt{\mu(B')\mu(B)} \tag{10}$$

We have a shorter right-hand-side than the full mixing lemma simply because we know that this value could only be bigger. With some manipulation:

$$\mu(B')(\frac{1}{2}-\epsilon_0) \leq \lambda\sqrt{\mu(B')\mu(B)} \tag{11}$$

$$\mu(B') \leq \frac{\lambda^2\mu(B)}{(\frac{1}{2}-\mu(B))^2} \tag{12}$$

$$\leq \frac{\lambda^2\epsilon_0}{(\frac{1}{2}-\epsilon_0)^2} \tag{13}$$

So we've bounded our new error. We achieve our goal if

$$\frac{\lambda^2\epsilon_0}{(\frac{1}{2}-\epsilon_0)^2} \leq \epsilon \tag{14}$$

When does that happen? When

$$\lambda \leq \frac{\sqrt{\epsilon}(\frac{1}{2}-\epsilon_0)}{\sqrt{\epsilon_0}} \tag{15}$$

Where λ is our second-largest eigenvalue. What if λ is too large, though? Well, we can decrease it as needed by simply powering A, which squares (and so shrinks) λ. We can determine the constant t such that λ^t is appropriately small, and that t is about $\log(\frac{1}{\epsilon})$, our initial claim. □

5 Randomness-Efficient Error Reduction

Can we shrink error even more if we allow for some extra random bits? That is the motivation question behind this next theorem.

Theorem 4 (Randomness-Efficient Error Reduction)**.** *Given a* BPP *algorithm M that uses r random bits on input x, we can reduce the error to $\leq \epsilon$ using* $\log(\frac{1}{\epsilon})$ *runs of the original algorithm and $r + O(\log\frac{1}{\epsilon})$ random bits.*

Compare to the naïve algorithm's cost to get those bounds: still $\log(\frac{1}{\epsilon})$ runs, but it uses $r \cdot O(\log\frac{1}{\epsilon})$ random bits. So we reduce a multiplicative factor to an additive one. The basic method behind this is choosing a random neighbor. Here we see why expanders act like complete graphs: if every node is completely connected, then choosing a random neighbor is equivalent to choosing a new random string. Expanders have a much smaller degree, but choosing a random neighbor is "almost as good."

Our actual method will be to do a length $\log\frac{1}{\epsilon}$ walk, and for each node (indicating a new random string) re-run our original algorithm, and once again take the majority vote.

To prove this, we need a key lemma.

Lemma 2 (Key Lemma)**.** *Let P be a projection on those ρ for which M errs, then for any vector x:*

$$\|PAx\|_2 \leq \sqrt{\epsilon_0+\lambda^2}\|x\|_2. \tag{16}$$

Proof. Consider the representation of $x = x^{\|} + x^{\perp}$. Then by the triangle inequality:

$$\|PAx\|_2 \le \|PAx^{\|}\|_2 + \|PAx^{\perp}\|_2. \tag{17}$$

Using the facts that the uniform distribution is invariant under A and that the bad set is small, we get:

$$\|PAx^{\|}\|_2 = \|Px^{\|}\|_2 \le \sqrt{\epsilon}\|x^{\|}\|_2. \tag{18}$$

Using the facts that P is a projection (and so does not increase the 2-norm), and that $x^{\perp}$ contracts by at least λ, we get:

$$\|PAx^{\perp}\|_2 \le \|Ax^{\perp}\|_2 \le \lambda\|x^{\perp}\|_2.. \tag{19}$$

Substituting (18) and (19) back into the (17), we have:

$$\begin{aligned}
\|PAx\|_2 &\le \sqrt{\epsilon_0}\|x^{\|}\|_2 + \lambda\|x^{\perp}\|_2 && (20)\\
&= (\sqrt{\epsilon_0}, \lambda) \cdot \left(\|x^{\|}\|_2, \|x^{\perp}\|_2\right)^{\mathsf{T}} && (21)\\
&\le \sqrt{\epsilon_0 + \lambda^2}\|x\|_2 && \text{(Follows from Cauchy-Schwarz)}
\end{aligned}$$

□

5.1 Using the lemma

This lemma can be used to bound the error probability of M'. Consider the probability that M' errs. This is the same as the probability that at least half of the steps in the random walk fall in the set where M errs.

$$\begin{aligned}
\Pr[M' \text{ errs}] &= \Pr[\text{At least } \tfrac{t}{2} \text{ of } \rho \text{ fall in the set on which } M \text{ errs}] && (22)\\
&\le \sum_{B\subseteq[t],|B|\ge\frac{t}{2}} \Pr[(\forall i \in B) i^{\text{th}} \text{ step lies in the bad set for } M] && (23)\\
&= \sum_{B\subseteq[t],|B|\ge\frac{t}{2}} \|M_t A M_{t-1} A \dots M_2 A M_1 A M_0 u\|_1 && (24)\\
&\le \sum_{B\subseteq[t],|B|\ge\frac{t}{2}} \sqrt{2^r}\|M_t A \dots M_1 A M_0 u\|_2 && (25)\\
&\le \sum_{B\subseteq[t],|B|\ge\frac{t}{2}} \sqrt{2^r}\left(\sqrt{\epsilon_0+\lambda^2}\right)^{|B|}\|u\|_2 && (26)\\
&= \sum_{B\subseteq[t],|B|\ge\frac{t}{2}} \left(\sqrt{\epsilon_0+\lambda^2}\right)^{|B|} && (27)\\
&\le 2^t \cdot \left(\sqrt{\epsilon_0+\lambda^2}\right)^{\frac{t}{2}} && (28)\\
&= (4\sqrt{\epsilon_0+\lambda^2})^{\frac{t}{2}} \le \epsilon. && (29)
\end{aligned}$$

Line (23) is an upper bounds the actual probability because it is over counting the bad strings. This probability is rewritten as a product of matrices in (24) where

$$M_i = \begin{cases} P & i \in B \\ I & \text{otherwise.} \end{cases}$$

Line (24) follows as an equality with this reasoning: u is the initial state, M_0 "kills" the bad set, then A means we take another "step," and so we repeat. Line (26) follows from repeated applications of Lemma 2. A constant number of iterations can decrease $\sqrt{\epsilon_0 + \lambda^2}$ to less than $\frac{1}{4}$.

What if the left-hand-side of our conclusion is too large? What ways do we have to decrease that value? If ϵ_0 is too large, we can use our previous, deterministic error reduction to shrink it further. If λ^2 is too large, we can power our graph (as we did in our deterministic method in the case where λ was too large, and so shrink λ. These methods are sufficient to allow ϵ to be small.

If $4\sqrt{\epsilon_0 + \lambda^2} < 1$, then walking for $t = O(\log \frac{1}{\epsilon})$ steps will give error less than ϵ. This procedure uses r random bits to pick the starting vertex, and $\log d$ bits for each of the $\log \frac{1}{\epsilon}$ steps in the random walk for a total of $r + O(\log \frac{1}{\epsilon})$ random bits for M'.

6 Other Results

A stronger result which considers the variance of random walks is known a the Expander Chernoff Bound. It states that if you take a random walk that the fraction of times the walk lands in the bad set does not vary much from the expected number. Let X_i be an indicator variable that indicates the event that the i^{th} step lies in some set $B_i \subseteq V$. Then the probability that the walk varies from the the expected number of steps in the bad sets can be written as:

$$\Pr[\sum_i^t (X_i - \mu(B_i)) \geq at] \leq e^{-b(1-\lambda)a^2 t}. \tag{30}$$

where $a \geq 0$ and b is some universal constant. The probability that the walk varies from expected for a constant a decreases exponentially as t increases. This inequality reduces to the standard Chernoff Bound if G is a complete graph ($\lambda(G) = 0$ because $\text{rank}(G) = 1$ and all the X_i are independent).

Next Lecture We will consider space bounded derandomization.

7 Acknowledgments

Many thanks to Amanda Hittson, Nathan Collins, and Matthew Anderson, the authors of Lecture 13 for Spring 2010, and Tyson Williams, author of lecture 14 for Spring 2010.