Lecture 2 : Expander Graphs, Mixing lemma and Applications to randomness

Topics in Pseudo-randomness and Complexity Theory (Spring 2018)  
Rutgers University  
Swastik Kopparty  
Scribe: Danny Scheinerman, Harsha Tirumala

1 Expander graphs and the Mixing lemma

Recall that for a $d$-regular graph $G$ we associate the adjacency matrix, $A$, and the normalized adjacency matrix $M$. As $G$ is $d$-regular, we have $M = \frac{1}{d}A$. The eigenvalues of $M$ lie in $[-1,1]$. We used the fact last class that if all (except the first) eigenvalues are sufficiently less than one in absolute value, then a random walk on the graph approaches uniform quickly. Let’s formalize that.

Let $p : V \to \mathbb{R}$ be a probability distribution on the vertices. We will use $p$ interchangeably with the vector in $\mathbb{R}^n$ whose $i$-th coordinate is $p(i)$. Then, $M^j p$ is the distribution given by first choosing a random vertex according to $p$ and then doing a $j$-step random walk. Let $v_1, \ldots, v_n$ be an orthonormal basis for $M$ with $v_1 = (1/\sqrt{n}, \ldots, 1/\sqrt{n})$. The eigenvalues are $1 = \lambda_1, \lambda_2, \ldots, \lambda_n$. Then we can write

$$M^j p = \sum_{i=1}^{n} \alpha_i M^j v_i$$

$$= \sum_{i=1}^{n} \alpha_i \lambda_i^j v_i$$

$$= \alpha_1 v_1 + \sum_{i \geq 2} \alpha_i \lambda_i^j v_i$$

Thus we can conclude that $\|M^j p - \alpha_1 v_1\|_2^2 \ll \sum_{i=1}^{n} \alpha_i^2 \cdot \max_{i \geq 2} |\lambda_i|^{2j} \leq \max_{i \geq 2} |\lambda_i|^{2j}$.

Now let’s characterize a class of graphs that have $\max_{i \geq 2} |\lambda_i|$ small and thus we expect the random walk to approach uniform quickly.

Definition 1. A $\lambda$-expander graph is a regular graph for which all eigenvalues (but one) of the normalized adjacency matrix are at most $\lambda$ in absolute value.

Theorem 2. For all $d \geq 5$, for all $n$ sufficiently large there exists a $d$-regular $1/2$-expander graph.

This is challenging to prove. We may prove it later, but we will use this often. In fact more is true:

Theorem 3. Let $d \geq 5$. A random $d$-regular graph is a $1/2$-expander graph with high probability.
Theorem 4. For all \( d \geq 5 \), for all \( n \) sufficiently large there exists a strongly explicit \( d \)-regular \( 1/2 \)-expander graph.

Let’s be explicit about what “explicit” means.

Definition 5. A graph is explicit if given \( n \) in time \( \text{poly}(n) \) we can compute an adjacency matrix for the graph.

Definition 6. A graph is strongly explicit if given \( n, i \in [n] \) and \( j \in [n] \) we can find the \( j \)-th neighbor of \( i \) in time \( \text{poly}(\log(n)) \).

This guarantee is important since the size of the input is \( \log n \) bits and thus, the \( \text{poly}(\log n) \) time complexity is a poly-time algorithm for establishing the \( j \)-th neighbor of vertex \( i \) “on-demand”

Example: Let \( V = \mathbb{F}_2^{2n} \). That is, the vertex set of our graph is pairs of points \((x, y)\) with each coordinate lying in the prime field \( \mathbb{F}_p \). Let \( S = \left\{ \left( \begin{array}{cc} 1 & 1 \\ 0 & 1 \end{array} \right), \left( \begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \right), \left( \begin{array}{cc} 1 & -1 \\ 0 & 1 \end{array} \right), \left( \begin{array}{cc} 1 & 0 \\ -1 & 1 \end{array} \right) \right\} \). For each \((x, y) \in V\) join the vertex \( A(x, y) \) for each \( A \in S \). The resulting graph, \( G \), is 4-regular (note that \( S \) is closed under inverses). It turns out that \( G \) is a \( \lambda \)-expander graph with \( \lambda < 1 \).

If \( G \) is a \( d \)-regular, \( 1/2 \)-expander graph then the random walk starting at a fixed vertex is \( o(1) \) close to uniform after \( O(\log n) \) steps. By “close” we mean \( L_1 \) or statistical distance:

Definition 7. Given distributions \( p : V \to \mathbb{R} \) and \( q : V \to \mathbb{R} \), we define the distance \( \Delta(p, q) = \sum_{x \in V} |p(x) - q(x)| = \|p - q\|_1 \). We call this distance the \( L_1 \) or statistical distance.

This distance metric has the following nice property. If \( \Delta(p, q) < \epsilon \) then for any \( E \subseteq V \), \( |p(E) - q(E)| < \epsilon \). This follows since \( |p(E) - q(E)| = \left| \sum_{x \in E} p(x) - q(x) \right| \leq \sum_{x \in E} |p(x) - q(x)| \leq \sum_{x \in V} |p(x) - q(x)| = \Delta(p, q) \), where we have used the triangle inequality to establish the first inequality. This means that if \( p \) and \( q \) are \( \epsilon \)-close in statistical distance and we know that \( p(E) \) is small then \( q(E) \) is small (up to an added \( \epsilon \)).

So let \( u = \alpha_1 v_1 = (1/n, 1/n, \ldots, 1/n) \) be the uniform vector. We have that \( \|M^2p - u\|_2^2 \leq \frac{1}{22} < \frac{1}{n^{10}} \). So it is close in \( L_2 \) distance. To establish statistical distance we use the Cauchy-Schwarz inequality: \( \sum_{i=1}^{n} a_i b_i \leq \left( \sum_{i=1}^{a_i^2} \right)^{1/2} \left( \sum_{i=1}^{b_i^2} \right)^{1/2} \). One useful inequality that follows from this arises from letting \( b_i = 1 \) for each \( i \). Then we have \( \sum_{i=1}^{n} a_i \leq \left( \sum_{i=1}^{a_i^2} \right)^{1/2} \sqrt{n} \). Using this we have \( \|M^2p - u\|_1 \leq \|M^2p - u\|_2 \sqrt{n} \leq O\left( \frac{1}{n^{5/2}} \right) \).

Theorem 8 (Expander mixing lemma). Let \( G \) be a \( d \)-regular \( \lambda \)-expander graph. Let \( A, B \subseteq V \) be two sets of vertices (possibly overlapping). Let \( e(A, B) = |\{(a, b) \in E(G) : a \in A, b \in B\}| \). That
is $e(A, B)$ is the number of edges joining a vertex in $A$ to a vertex in $B$. This is standard notation. Then
\[
\left| e(A, B) - \frac{d}{n} |A||B| \right| \leq \lambda d \sqrt{|A||B|}.
\]

Note that the estimate $e(A, B) \approx \frac{d}{n} |A||B|$ is what one would expect for a random $d$-regular graph on $n$ vertices.

**Proof.** Let $\mathbf{1}_A : V \to \mathbb{R}$ be the indicator function of the set $A$. Similarly, define $\mathbf{1}_B$. We can write the eigen-decomposition of these functions: $\mathbf{1}_A = \sum_{i=1}^{n} \alpha_i v_i$ and $\mathbf{1}_B = \sum_{i=1}^{n} \beta_i v_i$. As before $\lambda_i$ is the eigenvalue associated to $v_i$ and $\lambda_1 = 1$. Furthermore, note that $\alpha_1 = \langle \mathbf{1}_A, v_1 \rangle = |A|/\sqrt{n}$ and similarly $\beta_1 = |B|/\sqrt{n}$. Then we can express $e(A, B)$ as follows:
\[
e(A, B) = \sum_{i,j} \mathbf{1}_A(i) \mathbf{1}_B(j) A_{ij}
= d \langle \mathbf{1}_A, M \mathbf{1}_B \rangle
= d \left( \sum_{i=1}^{n} \alpha_i v_i, \sum_{j=1}^{n} \beta_j \lambda_j v_j \right)
= d \sum_{i=1}^{n} \alpha_i \beta_i \lambda_i
\quad \text{(Recall that $\langle v_i, v_j \rangle = 0$ if $i \neq j$)}
= d \left( \alpha_1 \beta_1 \lambda_1 + \sum_{i \geq 2} \alpha_i \beta_i \lambda_i \right)
= d \frac{|A||B|}{n} + d \sum_{i \geq 2} \alpha_i \beta_i \lambda_i
\]

So we have our main term. We just need to bound the right hand term:
\[
\left| \sum_{i \geq 2} \alpha_i \beta_i \lambda_i \right| \leq d \left( \sum_{i \geq 2} |\alpha_i \beta_i| \right) \lambda
\leq d \lambda \left( \sum_{i \geq 2} \alpha_i^2 \right)^{1/2} \left( \sum_{i \geq 2} \beta_i^2 \right)^{1/2}
\quad \text{(Using Cauchy-Schwarz)}
= d \lambda |A||B|
\]
as desired.

\[\square\]

## 2 Limits of expansion

In this section, we will establish the limits of expansion that can be achieved given a $d$-regular, $\lambda$-expander graph. Smaller values for the second-largest eigenvalue (absolutely speaking) of a matrix size
lead to stronger guarantees on the mixing nature of the graph.

**Exploiting expansion using graph power**

**Definition 9.** Given a multigraph $G$ with adjacency matrix $A$, the $t$-th power of $G$ is the multigraph on the same vertex set $V(G)$ with the adjacency matrix $A^t$.

The edge count of the vertex pair $(i,j) \in V(G) \times V(G)$ in $G^t$ equals the number of walks of length $t$ from $i$ to $j$ in $G$. Note that self loops are treated as single outgoing/incoming edge, i.e. $A_{i,i}$ is the number of self loops (not twice the number of self loops which may also seem natural).

**Lemma 10 (Graph power expansion).** If $G$ is a $d$-regular, $\lambda$-expander graph, then $G^t$ is a $d^t$-regular, $\lambda^t$-expander graph.

*Proof.* It is easy to see that if $G$ is $d$-regular, $G^t$ is $d^t$-regular. So, let us look at the expansion parameters. Let $G$ have eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ corresponding to eigenvectors $v_1, v_2, \ldots, v_n$ respectively. Then for all $i \in [n]$ we have:

\[
M v_i = \lambda_i v_i \\
\Rightarrow M^t v_i = \lambda_i^t v_i \\
\Rightarrow A^t v_i = (d \lambda_i)^t v_i
\]

That is, $G^t$ has the same eigenvectors as $G$ with eigenvalues $\lambda_1^t, \lambda_2^t, \ldots, \lambda_n^t$ if $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues for the normalized adjacency matrix $M$.

Starting with a $d \geq 5$, 1/2-regular expander graph (existence guaranteed by theorem 2), we can construct a $d$-regular, $1/d^{0.1}$-expander graph using the graph power expansion property discussed above.

**Corollary 11.** For all $d \geq 5$, for all sufficiently large $n$, there exist $d$-regular $1/d^{0.1}$-expander graph.

**An $\Omega(1/\sqrt{d})$ expansion limit for graph derivatives**

Let $G$ be a $d$-regular, $\lambda$-expander graph with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. Also, let $\lambda = \max_{2 \leq i \leq n} |\lambda_i|$. We will see that graph power expansion has an $\Omega(1/d)$ expansion limit given a $d$-regular $\lambda$-expander graph.

Consider the graph $G^2$ with the adjacency matrix $A^2$. The matrix entry $A^2(i,j)$ equals the number of walks of length 2 between vertices $i,j \in G$. The eigenvalues of $A^2$ are $d^2 \lambda_1, d^2 \lambda_2, \ldots, d^2 \lambda_n$.

**Fact 12.** The trace of any matrix $H$ equals the sum of the eigenvalues of $H$. 

4
The above fact will help establish a lower bound on the second-largest eigenvalue of $G$ as follows:

$$Tr[A^2] = \sum_{i=1}^{n} \lambda_i(A^2)$$

where $\lambda_i(A^2)$ represents the i-th eigenvalue of the matrix $A^2$.

$$\Rightarrow Tr[A^2] = \sum_{i=1}^{n} (d\lambda_i)^2$$

$$\leq d^2 + (n - 1)(d\lambda)^2$$

$$\Rightarrow \lambda \geq \sqrt{\frac{dn - d^2}{(n-1)d^2}}$$

Since we think of $d \ll n$, we have:

$$\lambda \geq \Omega(1/\sqrt{d})$$

The above analysis reveals the extent of expansion we can hope to "squeeze" out of a given $d$-regular expander.

**Theorem 13.** Ramanujan graphs achieve $\lambda \geq \sqrt{d}$.

### 3 Application : Randomness in computing

In this section, we will see how the mixing properties of expanders can be used to bring down randomness requirements of randomized algorithms.

Suppose we have a function $f : \{0,1\}^n \rightarrow \{0,1\}$ and we are given a randomized algorithm $A$ with the following guarantee:

$$\forall x \in \{0,1\}^n \quad Pr[A(x,r) = f(x)] \geq 0.9$$

i.e. the randomized algorithm $A$ has an error probability of 0.1 at most. Typically we want a stronger guarantee on the error rate : we want the error rate to be a quickly decreasing function in the size of the input. We look at how much randomness is used up to provide this guarantee for evaluating $f$.

**Majority Polling : "Brute force" use of randomness**

A simple and natural method to lower the error rate of evaluating $f(x)$ is to run the algorithm $A(x, r)$ for uniformly and independently selected random runs $r_1, r_2, \ldots, r_t \in \{0,1\}^m$ and to return the majority poll returned by the selection of runs. The idea is that it is less likely that a majority of the runs will fail compared to an individual run. The analysis of the error rate follows:

Let $Y_i$ be the indicator variable for $A(x, r_i) \neq f(x)$. i.e. $Y_i = 1_{A(x,r_i)\neq f(x)}$. From the guarantee that error rate of $A$ is at most 0.1, we have:

$$\forall i \in [t] \quad Pr[Y_i = 1] \leq 0.1$$
\[ \Rightarrow \mathbb{E}(\sum_{i=1}^{t} Y_i) = \sum_{i=1}^{t} \mathbb{E}(Y_i) \leq 0.1t \]

where the above equality follows from the linearity of expectation principle. The majority polling algorithm fails when more than half fraction of the runs result in errors i.e. when \( \sum_{i=1}^{t} Y_i \geq 0.5t \). We will use the Chernoff bound to get an error-bound for majority polling as follows:

\[
\Pr[|\sum_{i=1}^{t} Y_i - \mathbb{E}(\sum_{i=1}^{t} Y_i)| > \epsilon t] \leq e^{-\Omega(\epsilon^2 t)}
\]

where \( \epsilon = 0.4 \) corresponds to the LHS representing error case probabilities.

If the randomized algorithm runs in time \( T \) and uses \( m \) random bits per run, then in time \( O(tT) \) and with \( O(mt) \) random bits, we can reduce the error probability of evaluating \( f \) to \( e^{-\Omega(t)} \).

Randomness is an expensive resource. So, although the majority polling method provides the required guarantee on the error rate, we look for more efficient randomized algorithms.

**Bad proposal : naive ”Leader” neighborhood sampling**

1. Select a ”leader” candidate \( r_0 \in \{0,1\}^m \) uniformly at random.
2. Pick the next \( t \) strings \( \{r_1,r_2 \ldots,r_t\} \in \{0,1\}^m \) according to lexicographic order.
3. Run the algorithm \( A \) on each of these \( t \) strings.
4. Return \( A*(x) = \text{Majority } (\{A(x,r_i)\}_{i=1}^{t}) \)

Analysis :
Fixing \( x \), let \( B \) represent the set of bad strings for \( x \):

\[ B = \{ r : A(x, r) \neq f(x) \} \]

Now, let \( B^* \) be the set of bad ”leaders” for \( x \) i.e. those ”leaders” with \( \geq d/2 \) bad strings for \( x \):

\[ B^* = \{ s : A^*(x, s) \neq f(x) \} \]

Note that depending on the distribution of \( B \), it is possible the set \( B^* \) is as large as the set \( B \) i.e. it has \((0.1)2^m\) elements. This happens when all the bad strings for \( x \) occur in lexicographic succession.

The above algorithm needs randomness only to select the leader - and hence uses \( m \) bits of randomness. The time complexity of this algorithm is \( O(tT) \). The error rate in the worst case is 0.1, however, and thus this algorithm is not good enough.

One way of sampling the random runs efficiently is to pick a ”leader” run uniformly at random and then pick a group of runs deterministically based on the choice of the leader (eg: pick ”leader” randomly and then the next \( t \) consecutive binary strings in \( \{0,1\}^m \)).
**Good proposal : Expander-based "Leader" neighborhood sampling**

We will show that the mixing property of expander can be leveraged to pick robust neighborhoods for sampling runs.

1. Draw a $d$-regular, $1/d^{0.1}$-expander graph $G$ on the vertex set $\{0, 1\}^m$.
2. Pick a leader $s \in \{0, 1\}^m$ uniformly at random.
3. Let $N(s) = \{r_1, r_2, \ldots, r_d\} \in \{0, 1\}^m$ be the neighbors of $s$ in $G$.
4. Run $A$ on the set $N(s)$.
5. Output $A^*(x, s)$ as the majority of $A(x, r_i)$ where $i \in [d]$.

**Analysis :**
Let $B$ and $B^*$ represent the set of bad strings for $x$ and the set of bad "leaders" for $x$ respectively.

**Claim 14.** $|B^*| \leq \frac{10}{d^{0.2}} |B|$

**Proof.** Since $B^*$ is the set of "bad leaders" for $x$, we have:

$$e(B, B^*) \geq \frac{d}{2} |B|$$

Applying the expander mixing lemma on the sets $B, B^*$ we have :

$$|e(B, B^*) - \frac{d}{2m} |B||B^*| \leq d\lambda \sqrt{|B||B^*|}$$

$$\Rightarrow \frac{d}{2} |B| \leq e(B, B^*) \leq \frac{d}{2m} |B||B^*| + d\lambda \sqrt{|B||B^*|}$$

$$\Rightarrow |B^*| \left(\frac{1}{2} - \frac{|B|}{2m}\right) \leq \lambda \sqrt{|B||B^*|}$$

$$\Rightarrow \sqrt{|B^*|} \leq \frac{\lambda \sqrt{|B|}}{\frac{1}{2} - \frac{|B|}{2m}} \leq \frac{\lambda}{0.4} \sqrt{|B|}$$

$$\Rightarrow |B^*| \leq \frac{\lambda^2}{0.16} |B| \leq \frac{10}{d^{0.2}} |B|$$

So, when $t = d$, this algorithm runs in time $tT + poly(m)$ and uses $m$ bits of randomness to give an error probability $O(\frac{1}{d^{0.2}})$. Note that the $poly(m)$ time requirement is needed to reveal the structure of the strongly explicit expander graph $G$. 

---

7
"Random walks on expanders mix quickly"

We will now show that using a random walk on the expander graph to select the \( t \) runs \( G \) for \( A^* \) results in even better error bounds for evaluating \( f \).

Concretely, consider a \( d \)-regular, \( \lambda \)-expander graph \( G \) on the vertex set \( \{0, 1\}^m \). Pick a leader \( r_0 \in \{0, 1\}^m \) uniformly at random and take a random walk \( r_0, r_1, \ldots, r_t \) of length \( t \) in graph \( G \) from \( r_0 \). The vertices on the walk are the strings that we run \( A^* \) on. Output the majority of \( A^*(x, r_i) \) as the estimate for \( f(x) \) where \( i \in [t] \).

This algorithm runs in time \( t(poly(m) + T) \) and uses \( m + t \log(d) \) bits of randomness. Further analysis follows in the next lecture.