Spectral Graph Theory Lecture 12

Expander Codes

Daniel A. Spielman October 9, 2009

12.1 Overview

In this lecture, I will show how Zemor [Zem01] used expander graphs to construct asymptotically good error-correcting codes and decode them efficiently.

12.2 Bipartite Expander Graphs

Our construction of error-correcting codes will exploit bipartite expander graphs (as these give a much cleaner construction than the general case). Let's begin by examining what a bipartite expander graph should look like. It's vertex set will have two parts, U and V, each having n vertices. Every vertex will have degree d, and every edge will go from a vertex in U to a vertex in V.

In the same way that we view ordinary expanders as approximations of complete graphs, we will view bipartite expanders as approximations of complete bipartite graphs[1]. That is, if we let $K_{n,n}$ denote the complete bipartite graph, then we want a d-regular bipartite graph G such that

$$(1-\epsilon)\frac{d}{n}K_{n,n} \preccurlyeq G \preccurlyeq (1+\epsilon)\frac{d}{n}K_{n,n}.$$

As the eigenvalues of the Laplacian of $\frac{d}{n}K_{n,n}$ are 0 and $2d$ with multiplicity 1 each, and d otherwise, this means that we want a d-regular graph G whose Laplacian spectrum satisfies

$$\lambda_1 = 0, \quad \lambda_{2n} = 2d, \quad \text{and } |\lambda_i - d| \leq \epsilon d, \text{for all } 1 < i < 2n.$$

We can obtain such a graph by taking the *double-cover* of an ordinary expander graph.

Definition 12.2.1. *Let $G = (V, E)$ be a graph. The double-cover of G is the graph with vertex set $V \times \{0, 1\}$ and edges*

$$((u, 0), (v, 1)), \quad \textit{for } (u, v) \in E.$$

It is easy to determine the eigenvalues of the double-cover of a graph.

Proposition 12.2.2. *Let H be the double-cover of G. Then, for every eigenvalue λ_i of the Laplacian of G, H has a pair of eigenvalues,*

$$\lambda_i \quad \textit{and} \quad 2d - \lambda_i.$$

[1] The complete bipartite graph contains all edges between U and V

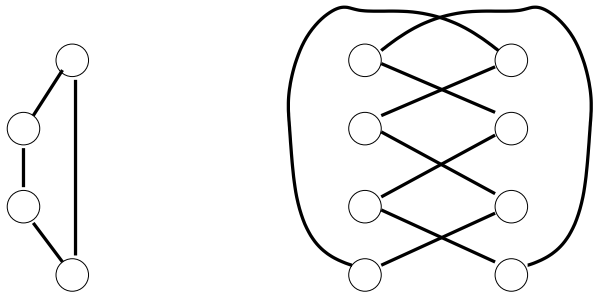

Figure 12.1: The cycle on 4 vertices, and its double-cover

The easiest way to prove this is to observe that if A is the adjacency matrix of G, then the adjacency matrix of H looks like

$$\begin{pmatrix} \mathbf{0} & A \\ A & \mathbf{0} \end{pmatrix}.$$

Our analysis of error-correcting codes will exploit the following theorem, which is analogous to Theorem 10.2.1.

Theorem 12.2.3. *Let $G = (U \cup V, E)$ be a d-regular bipartite graph that ϵ-approximates $\frac{d}{n}K_{n,n}$. Then, for all $S \subseteq U$ and $T \subseteq V$,*

$$\left| |E(S,T)| - \frac{d}{n}|S|\,|T| \right| \leq \epsilon d \sqrt{|S|\,|T|}.$$

Proof. First show that $\left\| L_G - \frac{d}{n} L_{K_{n,n}} \right\| \leq \epsilon d$. The rest of the proof is the same as that of Theorem 10.2.1. □

Let $G(S \cup T)$ denote the graph induced on vertex set $S \cup T$. We use the following simple corollary of Theorem 12.2.3.

Corollary 12.2.4. *Let $|S| = \sigma n$ and let $|T| = \tau n$. The average degree of vertices in $G(S \cup T)$ is at most*

$$\frac{2d\sigma\tau}{\sigma+\tau} + \epsilon d.$$

Proof. The average degree of a graph is twice its number of edges, divided by the number of vertices. In our case, this is at most

$$\frac{2d}{n}\frac{|S|\,|T|}{|S|+|T|} + 2\epsilon d \frac{\sqrt{|S|\,|T|}}{|S|+|T|}.$$

The left-hand term is

$$\frac{2d\sigma\tau}{\sigma+\tau},$$

and the right-hand term is at most

$$\epsilon d.$$

□

12.3 Building Codes

Our construction of error-correcting codes will require two ingredients: a d-regular bipartite expander graph G on $2n$ vertices, and a linear error correcting code C_0 of length d. We will combine these to construct an error correcting code of length dn. We think of the code C_0 as being a small code that drives the construction. This is reasonable as we will keep d a small constant while n grows.

In our construction of the code, we associate one bit with each edge of the graph. As the graph has dn edges, this results in dn bits, which we label $y_1, \ldots, y_{dn}$. We now describe the code by listing the linear constraints its codewords must satisfy. Each vertex requires that the bits on its attached edges resemble a codeword in the code C_0. That is, each vertex should list its attached edges in some order (which order doesn't matter, but it should be fixed). As a vertex has d attached edges, it is easy to require that the d bits on these edges are a codeword in the code C_0.

Let r_0 be the rate of code C_0. This means that the space of codewords has dimension $r_0 d$. But, since C_0 is a linear code, it means that its codewords are exactly the the vectors that satisfy some set of $d(1-r_0)$ linear equations. As there are $2n$ vertices in the graph, the constraints imposed by each vertex impose $2nd(1-r_0)$ linear constraints on the dn bits. Thus, the vector space of codewords that satisfy all of these constraints has dimension at least

$$dn - 2dn(1-r_0) = dn(2r_0 - 1),$$

and the code we have constructed has rate at least

$$r = 2r_0 - 1.$$

So, this rate will be a non-zero constant as long as $r_0 > 1/2$.

For the rest of the lecture, we will let C denote the resulting expander code.

12.4 Encoding

Note that I have described the set of codewords, but have not said how one should encode. I did that on purpose. As the code is linear, it is relatively easy to find a way to encode it. In particular, one may turn the above description of the code into a matrix M with dn columns and $2dn(1-r_0)$ rows such that the codewords are precisely those $\boldsymbol{y}$ such that

$$M\boldsymbol{y} = \mathbf{0}.$$

So, the codewords form a vector space of dimension $dn(2r_0 - 1)$, and so there is a matrix N with $dn(2r_0 - 1)$ columns and dn rows for which the codewords are precisely the vectors $N\boldsymbol{x}$, for $\boldsymbol{x} \in \{0,1\}^{dn(2r_0-1)}$. In fact, there are many such matrices N, and they are called *generator matrices* for the code. Such a matrix N may be computed from M by elementary linear algebra.

As the matrix N will be dense, this leads to an encoding algorithm that takes time $\Theta((dn)^2)$. Of course, one would prefer to encode them in time $O(dn)$. Using Ramanujan expanders and the Fast

Fourier Transform over the appropriate groups, Lafferty and Rockmore [LR97] reduced the time for encoding to $O(d^2 n^{4/3})$. Spielman [Spi96] modifies the code construction to obtain codes with similar performance that may be encoded in linear time.

12.5 Minimum Distance

We will now see that if C_0 is a good code, then C has large minimum distance. Let $\delta_0 d$ be the minimum distance of the code C_0. You should think of δ_0 as being a constant.

Theorem 12.5.1. *If $\epsilon \leq \delta_0/2$, then the minimum relative distance δ of C satisfies*

$$\delta \geq \delta_0^2/2.$$

Proof. As we saw last lecture, it suffices to prove that C has no codewords of small Hamming weight. To this end, we identify a codeword with the set of edges on which its bits are 1. Let F be such a set of edges, and let $|F| = \phi dn$. As the minimum distance of C_0 is $\delta_0 d$, every vertex v that is attached to an edge of F must be attached to at least $\delta_0 d$ edges of F. Let S be the subset of vertices of U adjacent to edges in F, and let T be the corresponding subset of V. We have just argued that every vertex in $G(S \cup T)$ must have degree at least $\delta_0 d$, and so in particular the average degree of $G(S \cup T)$ is at least $\delta_0 d$.

We may also use this fact to see that

$$|S|, |T| \leq \frac{|F|}{\delta_0 d}.$$

Set $\sigma = |S|/n$ and $\tau = |T|/n$, so the previous inequality becomes

$$\sigma, \tau \leq \frac{\phi}{\delta_0}.$$

So, Corollary 12.2.4 tells us that the average degree of $G(S \cup T)$ is at most

$$d\frac{\phi}{\delta_0} + \epsilon d.$$

Combining the upper and lower bounds on the average degree of $G(S \cup T)$, we obtain

$$\delta_0 d \leq d\frac{\phi}{\delta_0} + \epsilon d,$$

which implies

$$\delta_0(\delta_0 - \epsilon) \leq \phi.$$

The assumption $\epsilon \leq \delta_0/2$ then yields

$$\phi \geq \delta_0^2/2.$$

As we assumed that F was the set of edges corresponding to a codeword and that $|F| = \phi dn$, we have shown that the minimum relative distance of C is at least $\delta_0^2/2$. □

12.6 Decoding

We will convert an algorithm that corrects errors in C_0 into an algorithm for correcting errors in C. The construction is fairly simple. We first apply the decoding algorithm at every vertex in U. We then do it at every vertex in V. We alternate in this fashion until we produce a codeword.

To make this more concrete, assume that we have an algorithm A that corrects up to $\delta_0 d/2$ errors in the code C_0. That is, on input any word $\boldsymbol{r} \in \{0,1\}^d$, A outputs another word in $\{0,1\}^d$ with the guarantee that if there is a $\boldsymbol{c} \in C_0$ such that $\mathrm{dist}(\boldsymbol{c}, \boldsymbol{r}) \leq \delta_0 d/2$, then A outputs $\boldsymbol{c}$. We then apply the transformation A independently to the edges attached to each vertex of U. We then do the same for V, and then work back again.

We will prove that under the right conditions this algorithm will correct up to $\delta_0^2 dn/18$ errors in at most $\log_{4/3} n$ iterations. The idea is to keep track of which vertices are attached to edges that contain errors, rather than keeping track of the errors themselves. We will exploit the fact that any vertex that is attached to few edges in error will correct those errors. Let S be the set of vertices attached to edges in error after a U-decoding step. We will show that the set T of vertices attached to edges in error after the next V-decoding step will be much smaller.

Lemma 12.6.1. *Assume that $\epsilon \leq \delta_0/3$. Let $F \subset E$ be a set of edges, let S be the subset of vertices in U attached to edges in F and let T be the subset of vertices in V attached to at least $\delta_0 d/2$ edges in F. If*

$$|S| \leq \delta_0 n/9,$$

then

$$|T| \leq \frac{3}{4} |S| .$$

Proof. Let $|S| = \sigma n$ and $|T| = \tau n$. We have $|F| \geq (\delta_0 d/2)\,|T|$. As the average degree of $G(S \cup T)$ is twice the number of edges in the subgraph divided by the number of vertices, it is at least

$$\frac{\delta_0 d\,|T|}{|S| + |T|} = \frac{\delta_0 d\tau}{\sigma + \tau}.$$

Applying Corollary 12.2.4, we find

$$\frac{\delta_0 d\tau}{\sigma + \tau} \leq \frac{2d\sigma\tau}{\sigma + \tau} + \epsilon d.$$

This implies

$$\delta_0 \tau \leq 2\sigma\tau + \epsilon(\sigma + \tau),$$

which becomes

$$\tau \leq \frac{\epsilon\sigma}{\delta_0 - 2\sigma - \epsilon}.$$

Recalling that $\sigma \leq \delta_0/9$ and $\epsilon \leq \delta_0/3$, we obtain

$$\tau \leq \sigma \frac{\delta_0/3}{\delta_0(4/9)} \leq \frac{3}{4}\sigma.$$

□

Lemma 12.6.2. *Assume that $\epsilon \leq \delta_0/3$. Let F be the set of edges in error after a U-decoding step, and let S be the set of vertices in U attached to F. Now, perform a V-decoding step and let T be the set of vertices in V attached to edges in error afterwards. If*

$$|S| \leq \delta_0 n/9,$$

then

$$|T| \leq \frac{3}{4}|S|.$$

Proof. Every vertex in V that outputs an error after the V-decoding step must be attached to at least $\delta_0 d/2$ edges of F. Moreover, each of these edges is attached to a vertex of S. Thus, the lemma follows immediately from Lemma 12.6.1. □

Theorem 12.6.3. *If $\epsilon \leq \delta_0/3$, then the proposed decoding algorithm will correct every set of at most*

$$\frac{\delta_0^2}{18} dn$$

errors.

Proof. Let F denote the set of edges that are initially in error. Let S denote the set of vertices that output errors after the first U-decoding step. Every vertex in S must be adjacent to at least $\delta d/2$ edges in F, so

$$|F| \leq \frac{\delta_0^2}{18} dn \quad \Longrightarrow \quad |S| \leq \frac{|F|}{\delta_0 d/2} \leq \delta_0 n/9.$$

After this point, we may apply Lemma 12.6.2 to show that the decoding process converges in at most $\log_{4/3} n$ iterations. □

12.7 Historical Notes

Gallager [Gal63] first used graphs to construct error-correcting codes. His graphs were also bipartite, with one set of vertices representing bits and the other set of vertices representing constraints. Tanner [Tan81] was the first to put the vertices on the edges. The use of expansion in analyzing these codes we pioneered by Sipser and Spielman [SS96].

The construction we present here is due to Zemor [Zem01], although he presents a tighter analysis. Improved constructions and analyses may be found in [BZ02, BZ05, BZ06, AS06].

References

[AS06] A. Ashikhmin and V. Skachek. Decoding of expander codes at rates close to capacity. *Information Theory, IEEE Transactions on*, 52(12):5475–5485, Dec. 2006.

[BZ02] A. Barg and G. Zemor. Error exponents of expander codes. *Information Theory, IEEE Transactions on*, 48(6):1725–1729, Jun 2002.

[BZ05] A. Barg and G. Zemor. Concatenated codes: serial and parallel. *Information Theory, IEEE Transactions on*, 51(5):1625–1634, May 2005.

[BZ06] A. Barg and G. Zemor. Distance properties of expander codes. *Information Theory, IEEE Transactions on*, 52(1):78–90, Jan. 2006.

[Gal63] R. G. Gallager. *Low Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.

[LR97] John D. Lafferty and Daniel N. Rockmore. Spectral techniques for expander codes. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 160–167, New York, NY, USA, 1997. ACM.

[Spi96] D.A. Spielman. Linear-time encodable and decodable error-correcting codes. *Information Theory, IEEE Transactions on*, 42(6):1723–1731, Nov 1996.

[SS96] M. Sipser and D.A. Spielman. Expander codes. *Information Theory, IEEE Transactions on*, 42(6):1710–1722, Nov 1996.

[Tan81] R. Michael Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981.

[Zem01] G. Zemor. On expander codes. *Information Theory, IEEE Transactions on*, 47(2):835–837, Feb 2001.