1 Expanders as approximations of the complete graph

The fact that random walks on expanders converge quickly to the uniform distribution says that, in some sense, an expander is a good approximation of a complete graph: In a complete graph (with a loop around every vertex), the uniform distribution is reached after one step of the random walk, no matter where we start from. In general, an expander cannot do this because it has bounded degree, but in certain situations we can think of an expander graph as a complete graph "plus" some error.

We know that it takes an expander about $1/(1-\lambda)$ steps to get within distance $1/2$ of the uniform distribution, starting from any vertex s. One imprecise way to state this is that at each step of the random walk, the expander behaves like a complete graph with probability $1-\lambda$, and like some other, "error graph" with the remaining probability.

Let us try to formalize this intuition now. We want to think of the adjacency matrix A of G as representing the adjacency martix J of the complete graph with probability $1-\lambda$ (all the entries of J have value $1/n$) plus some "error matrix" E. We write $A = (1-\lambda)J + E$. The reason E can be thought of as an "error matrix" is that E^t vanishes as t becomes large. More precisely we have the following theorem:

Theorem 1. *If G is an expander with adjacency matrix A and $A = (1-\lambda)J + E$, then for every vector $\mathbf{v}$, $\|\mathbf{v}E\| \leq \lambda\|\mathbf{v}\|$.*

Proof. Let's first assume that $\mathbf{v} \perp \mathbf{u}$. We write $\mathbf{v}$ in the basis $\mathbf{v}_1, \dots, \mathbf{v}_n$ as

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_n\mathbf{v}_n.$$

Since $\mathbf{v} \perp \mathbf{u}$, $\alpha_1 = 0$, and $\mathbf{v}E = \mathbf{v}A - (1-\lambda)\mathbf{v}J = \mathbf{v}A$ because $\mathbf{v}J = 0$. It follows that

$$\mathbf{v}A = \mathbf{v}A = \lambda_2\alpha_2\mathbf{v}_2 + \cdots + \lambda_n\alpha_n\mathbf{v}_n$$

and so

$$\|\mathbf{v}E\|^2 = \lambda_2^2\alpha_2^2 + \cdots + \lambda_n^2\alpha_n^2 \leq \lambda^2\|\mathbf{v}\|^2.$$

Now assume $\mathbf{v} \not\perp \mathbf{u}$. Then by rescaling we can assume the entries of $\mathbf{v}$ add up to one. We write

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_n\mathbf{v}_n$$

Then $\alpha_1\mathbf{v}_1 = \langle \mathbf{v}, \mathbf{v}_1\rangle\mathbf{v}_1 = \mathbf{u}$ — the uniform distribution, and

$$\mathbf{u}E = \mathbf{u}A - (1-\lambda)\mathbf{u}J = \mathbf{u} - (1-\lambda)\mathbf{u} = \lambda\mathbf{u}.$$

On the other hand, $(\mathbf{v}-\mathbf{u}) \perp \mathbf{u}$, so by the previous analysis we know that

$$(\mathbf{v}-\mathbf{u})E = (\mathbf{v}-\mathbf{u})A = \lambda_2\alpha_2\mathbf{v}_2 + \cdots + \lambda_n\alpha_n\mathbf{v}_n.$$

So, we have that

$$\mathbf{v}E = \mathbf{u}E + (\mathbf{v}-\mathbf{u})E = \lambda\mathbf{u} + (\mathbf{v}-\mathbf{u})A = \lambda\alpha_1\mathbf{v}_1 + (\lambda_2\alpha_2\mathbf{v}_2 + \cdots + \lambda_n\alpha_n\mathbf{v}_n)$$

and therefore

$$\|\mathbf{v}E\|^2 = \lambda^2\alpha_1^2 + \lambda_2^2\alpha_2^2 + \cdots + \lambda_n^2\alpha_n^2 \leq \lambda^2\|\mathbf{v}\|^2. \qquad \square$$

2 The zig-zag product

We will now show a construction that allows us to turn any connected, d-regular graph into an expanding family of d-regular graphs. The construction proceeds in stages, where at every stage the graph is replaced by a larger graph with the same degree and better expansion.

Given a graph G, one way to improve the value $\lambda(G)$ is to "square" the graph: To obtain the graph G^2, we take every path (u, v, w) of length 2 in G and replace it by an edge (u, w). Then that the adjacency matrix of G^2 is A^2, the square of the adjacency matrix of A. The eigenvalues of G^2 are also squares of the eigenvalues of A, so $\lambda(G^2) = \lambda(G)^2$.

Unfortunately, the process of squaring a graph also squares the degree: If G has degree d, then $G' = G^2$ has degree d^2. As we want our graphs to be d-regular, we need some way to reduce the degree. One way to do so is to replace every vertex in G' with a "cloud" of d^2 vertices which are interconnected by some d-regular graph H. This operation could undo the improvement in λ we obtained from the squaring.

It turns out that if the graph H is itself a sufficiently good expander, the loss in the value of λ is very small compared to the gain we obtain form the squaring.

This sounds like circular reasoning: In order to build an expander, we already need to have an expander H. However, the point is that the graph H has only a *constant* number of vertices, while the number of vertices in the big expander we are building continues to grow. So we have reduced the task of building a family of expander (as n grows) to the task of building a small graph on d^2 vertices with small $\lambda(H)$, which is a considerably easier task.

Analysis of the zig-zag product Instead of replacing every vertex of G' with a cloud connected by H, for the analysis it will be more convenient to deal with a slightly more involved construction called the "zig-zag product". Let G be a d-regular graph on n vertices and H be a d'-regular graph on d vertices. The *zig-zag product* $G \textcircled{z} H$ is a d'^2-regular graph on nd vertices obtained as follows. First, we replace each vertex of G by a "cloud" of d vertices connected via H. Then we put edges as follows: Starting from any vertex, first we take a step within each cloud H, then we take a step in G, then we take a step in the new cloud H.

Let A and B be the adjacency matrices of G and H respectively. Let us describe the adjacency matrix of $G \textcircled{z} H$. This is an $nd \times nd$ matrix whose rows and columns are indexed by pairs (u, i), where u is a vertex of G and i is a vertex of H. Let's think of these vertices as divided into n blocks, where the uth block consists of $(u, 1), (u, 2), \ldots, (u, d)$. The adjacency matrix of $G ⓩ H$ will have the form $\tilde{B}\tilde{A}\tilde{B}$, where $\tilde{B}$ represents the step on H and $\tilde{A}$ is the step on G.

What do the matrices $\tilde{A}$ and $\tilde{B}$ look like? Notice that $\tilde{B}$ describes a move on H on each cloud; so the matrix $\tilde{B}$ is a block-diagonal matrix, where each block contains the matrix B. The matrix $\tilde{A}$ represents a move on G between the clouds, which is a matching (and in particular a permutation) of the vertices of $G ⓩ H$. More precisely, if we number the edges incident to each vertex of G by $1, 2, \ldots, d$, then $\tilde{A}_{(u,i),(v,j)} = 1$ if the ith edge incident to u is the same as the jth edge incident to v, and 0 otherwise.

Theorem 2. *If $\lambda(G) = 1 - \varepsilon(G)$ and $\lambda(H) = 1 - \varepsilon(H)$, then $\lambda(G ⓩ H) \leq 1 - \varepsilon(G)\varepsilon(H)^2$.*

In the proof it will be useful to use the following characterization of $\lambda(G)$ for any graph G with adjacency matrix $A(G)$, which is proved in the same way as the formula for λ_2:

$$\lambda(G) = \max_{\|\mathbf{v}\|=1, \mathbf{v}\perp\mathbf{u}} |\mathbf{v}A(G)\mathbf{v}^{\mathrm{T}}|.$$

Proof. Using the above characterization of λ we obtain:

$$\lambda(G ⓩ H) = \max_{\|\mathbf{v}\|=1, \mathbf{v}\perp\mathbf{u}} |\mathbf{v}(\tilde{B}\tilde{A}\tilde{B})\mathbf{v}^{\mathrm{T}}| = \max_{\|\mathbf{v}\|=1, \mathbf{v}\perp\mathbf{u}} |(\mathbf{v}\tilde{B})\tilde{A}(\mathbf{v}\tilde{B})^{\mathrm{T}}|.$$

Let $\mathbf{v}$ be a vector perpendicular to the uniform distribution $\mathbf{u}$ on the nd vertices of $G ⓩ H$ with $\|\mathbf{v}\| = 1$. We will bound $|(\mathbf{v}\tilde{B})\tilde{A}(\mathbf{v}\tilde{B})^{\mathrm{T}}|$ using Theorem 1. We write $B = (1 - \lambda(H))J + E = \varepsilon(H)J + E$. Since each block of $\tilde{B}$ consists of a copy of B, we can write $\tilde{B} = \varepsilon(H)\tilde{J} + \tilde{E}$, where $\tilde{J}$, $\tilde{E}$ are block-diagonal matrices containing a copy of J and E in each block, respectively. Then:

$$\begin{aligned} |(\mathbf{v}\tilde{B})\tilde{A}(\mathbf{v}\tilde{B})^{\mathrm{T}}| &= |(\varepsilon(H)\mathbf{v}\tilde{J} + \mathbf{v}\tilde{E})\tilde{A}(\varepsilon(H)\mathbf{v}\tilde{J} + \mathbf{v}\tilde{E})^{\mathrm{T}}| \\ &\leq \varepsilon(H)^2|(\mathbf{v}\tilde{J})\tilde{A}(\mathbf{v}\tilde{J})^{\mathrm{T}}| + 2\varepsilon(H)|(\mathbf{v}\tilde{E})\tilde{A}(\mathbf{v}\tilde{J})^{\mathrm{T}}| + |(\mathbf{v}\tilde{E})\tilde{A}(\mathbf{v}\tilde{E})^{\mathrm{T}}|. \end{aligned}$$

We now bound each of these terms. First, notice that because $\tilde{J}$ is itself an adjacency matrix, $\mathbf{v}\tilde{J} \perp \mathbf{u}$ and therefore we also have $\mathbf{v}\tilde{E} \perp \mathbf{u}$. Also, because $\tilde{J}$ and $\tilde{E}$ are block-diagonal copies of J and E, we have $\|\mathbf{v}\tilde{J}\| \leq 1$ and $\|\mathbf{v}\tilde{E}\| \leq \lambda(H)$.

To bound $|(\mathbf{v}\tilde{J})\tilde{A}(\mathbf{v}\tilde{J})^{\mathrm{T}}|$, notice that $\mathbf{v}\tilde{J}$ is constant on each block. Let $\mathbf{v}'(u)$ be the value that $\mathbf{v}\tilde{J}$ takes in block u. If we think of $\mathbf{v}'$ as a vector with n entries, then $\|\mathbf{v}\tilde{J}\|^2 = d\|\mathbf{v}'\|^2$. Then

$$|(\mathbf{v}\tilde{J})\tilde{A}(\mathbf{v}\tilde{J})^{\mathrm{T}}| = \Big|\sum_{\text{edge } (u,w) \text{ in } G} \mathbf{v}'(u)\mathbf{v}'(w)\Big| = |\mathbf{v}'(dA)\mathbf{v}'^{\mathrm{T}}| \leq d\lambda(G)\|\mathbf{v}'\|^2 = \lambda(G)\|\mathbf{v}\tilde{J}\|^2 \leq \lambda(G).$$

For the other two terms, we use the Cauchy-Schwarz inequality, together with the fact that $\tilde{A}$ is a permutation matrix, so it does not affect norms:

$$\begin{aligned} |(\mathbf{v}\tilde{E})\tilde{A}(\mathbf{v}\tilde{J})^{\mathrm{T}}| &\leq \|\mathbf{v}\tilde{E}\tilde{A}\|\|\mathbf{v}\tilde{J}\| = \|\mathbf{v}\tilde{E}\|\|\mathbf{v}\tilde{J}\| \leq \lambda(H)\cdot 1, \\ |(\mathbf{v}\tilde{E})\tilde{A}(\mathbf{v}\tilde{E})^{\mathrm{T}}| &\leq \|\mathbf{v}\tilde{E}\tilde{A}\|\|\mathbf{v}\tilde{J}\| = \|\mathbf{v}\tilde{E}\|^2 \leq \lambda(H)^2. \end{aligned}$$

Substituting these inequalities we obtain:

$$|(\mathbf{v}\tilde{B})\tilde{A}(\mathbf{v}\tilde{B})^{\mathrm{T}}| \leq \varepsilon(H)^2\lambda(G) + 2\varepsilon(H)\lambda(H) + \lambda(H)^2 = 1 - \varepsilon(H)^2\varepsilon(G)$$

which proves the theorem, since $\mathbf{v}$ was an arbitrary vector perpendicular to $\mathbf{u}$ of norm 1. $\square$

3 Expanders via the zig-zag product

We now show how to turn an arbitrary d-regular graph G with $\lambda < 1$ (where d is sufficiently large, but constant) into a new, slightly larger graph, with $\lambda < 1/2$ and same degree.

The idea is to iterate the operations of squaring and zig-zag product. Squaring G increases the degree and decreases λ, while the zig-zag product decreases the degree but increases λ. With a careful choice of parameters we can arrange that at each step, the degree stays the same while λ keeps decreasing.

To apply the zig-zag product, we need a suitably chosen "cloud" graph H. You will prove the existence of this H in the homework.

Claim 3. *For every sufficiently large d, there exists a $\sqrt{d}$-regular graph H on d^2 vertices such that $\lambda(H) < 0.1$.*

Suppose now that G is a d-regular graph on n vertices with $\lambda(G) = 1 - \varepsilon$, where $\varepsilon < 0.5$. After squaring, we obtain the graph G^2, which has n vertices, is d^2-regular, and

$$\lambda(G^2) = (1-\varepsilon)^2 = 1 - (2-\varepsilon)\varepsilon) < 1 - 1.5\varepsilon.$$

The graph $G^2 ⓩ H$ has nd^2 vertices, is d-regular, and by Theorem 2, we have

$$\lambda(G^2 ⓩ H) < (1.5\varepsilon)(\varepsilon(H)^2) = 1.5 \cdot (0.9)^2\varepsilon < 1.2\varepsilon.$$

Let us summarize this analysis in the following table:

graph	vertices	degree	$1-\lambda$
G	n	d	ε
G^2	n	d^2	1.5ε
$G^2 ⓩ H$	nd^2	d	1.2ε

Let's now iterate this construction. As long as $1 - \lambda$ stays above $1/2$ throughout the process, after t steps we obtain a d-regular graph on nd^{2t} vertices with $1 - \lambda < (1.2)^t\varepsilon$. Initially G is connected, so $\varepsilon \geq 1/2(nd)^2$, and $1 - \lambda$ must drop under $1/2$ after at most $O(\log nd)$ iterations. The resulting graph has $n^{O((\log d)^2)}$ vertices, is d-regular, and has $\lambda < 1/2$.

4 Undirected paths in logarithmic space

The zig-zag product construction of expanders we just saw gives an elegant *deterministic* algorithm for undirected paths in logarithmic space. We sketch how this algorithm works.

The idea behind the algorithm is the following. Suppose that we could ensure that the component of G containing s is a d-regular expander, where d is constant and (say) $\lambda < 1/2$. As we saw in **??**, if we were to do a random walk starting at s of length $O(\log n)$, we have a nonzero probability of visiting *every* vertex connected to s. But such a random walk can be simulated deterministically in logarithmic space: At each step of the walk, we only need to remember which one of the d possible edges we took out of the vertex. Since d is constant and there will be only $O(\log n)$ vertices to remember, we only need logarithmic space.

In general, we cannot assume that G has any expansion properties. But now suppose we apply the zig-zag construction of expanders to the graph G (using a suitable H for the zig-zag product steps) to obtain a graph G'. If s and s' are in the same connected component of G, then after zig-zagging, this component will become an expanding component. G' has $\text{poly}(n)$ vertices and each vertex of G' represents some vertex of G, so a path from s to s' can be detected by running the deterministic logarithmic space path algorithm for expanding graphs. On the other hand, if s and s' are not connected in G, then their representatives in G' will fall into different components, so no path between s and s' can be detected in G'.

To realize this idea, we need to make sure that we can compute the graph G' from G in logarithmic space. Recall that initially G is a 4-regular graph, which can be made d-regular for some larger d by adding parallel edges. Since the graph H has *constant* size, it can be stored without any essential effect on the space complexity. (You can even think of H as hard-coded in the states of the Turing Machine.) It is convenient to represent graphs by their *rotation map*: This is a data structure that on input (u, i) outputs (v, j), where the ith edge incident to u is the same as the jth edge incident to v.

The rotation map of G' can be computed recursively (given the rotation maps of G and H). To do so it suffices to show that in both the squaring and the zig-zag product steps, the rotation map of the new graph can be computed in place using only a constant amount of extra scratch space. Since these steps are applied $O(\log n)$ times, the whole computation will use logarithmic space. We omit the exact details of the space-efficient computations of rotation maps, but they are not difficult.