Topics in Theoretical Computer Science March 20, 2014

Lecture 5

Lecturer: Ola Svensson *Scribes: Cijo Jose & James Newling*

1 Last week

In the last lecture we were introduced to ideas underlying spectral graph theory. In particular we saw the close connection between the expansion properties of a d-regular graph (as introduced in Lecture 3) and the eigenvalues of a graph's normalised adjacency matrix. Suppose that the n eigenvalues of a d-regular graph $\mathcal{G} = (V, E)$ are,

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n.$$

We saw that,

- $\lambda_1 = 1$,
- $\lambda_2 = 1$ if and only if $\mathcal{G}$ is disconnected, and
- $\lambda_n = -1$ if and only if $\mathcal{G}$ is bipartite.

We also proved the lower bound of Cheeger's Inequality,

$$\frac{1-\lambda_2}{2} \leq h(\mathcal{G}) \leq \sqrt{2(1-\lambda_2)},$$

where,

$$h(\mathcal{G}) = \min_{S \subset V} h(S),$$
$$h(S) = \frac{E(S, \bar{S})}{d \cdot \min(S, \bar{S})},$$

where $E(S, \bar{S})$ is the number edges which connect a vertex in S to a vertex in $\bar{S}$. Today we will prove the upper bound (the "difficult" direction), which can be interpreted as saying that if the normalised adjacency matrix M of $\mathcal{G}$ has a large second eigenvalue, then $\mathcal{G}$ can be split into two parts that are poorly connected to each other, i.e. $\mathcal{G}$ is a poor expander.

2 Proving the Upper Bound of Cheeger's Inequality

It will suffice to prove that there exists a set $S \subset V$ such that $h(S) \leq \sqrt{2(1-\lambda_2)}$, as $\forall S : h(\mathcal{G}) \leq h(S)$. We will prove this by 'averaging'[1] over a particular set of subsets of V which we expect to have low values of h, and show that this 'average' is bounded above by $\sqrt{2(1-\lambda_2)}$. Our choice for the set of subsets will now be motivated informally,

2.1 Sets S for which we expect $h(S)$ to be low.

Consider the second eigenvalue and a corresponding eigenvector of the normalised adjacency matrix M: $Mv_2 = \lambda_2 v_2$. Initialise each vertex in V to its corresponding value in v_2. The average value of the neighbours of a vertex of value s is then $\lambda_2 s$. If λ_2 is large (say 0.999) we then have that a vertex of value s has a neighbour-average of $0.999s \approx s$, and thus we expect neighbour values to be around s. Thus if we were to take S to be all vertices with v_2 value less than some threshold t, we would expect only vertices in S with a v_2 value close to t to have an edge from S to $\bar{S}$. This is illustrated in Figure 1, where t is represented by the green line.

[1]meaning to be made precise in Section 2.2

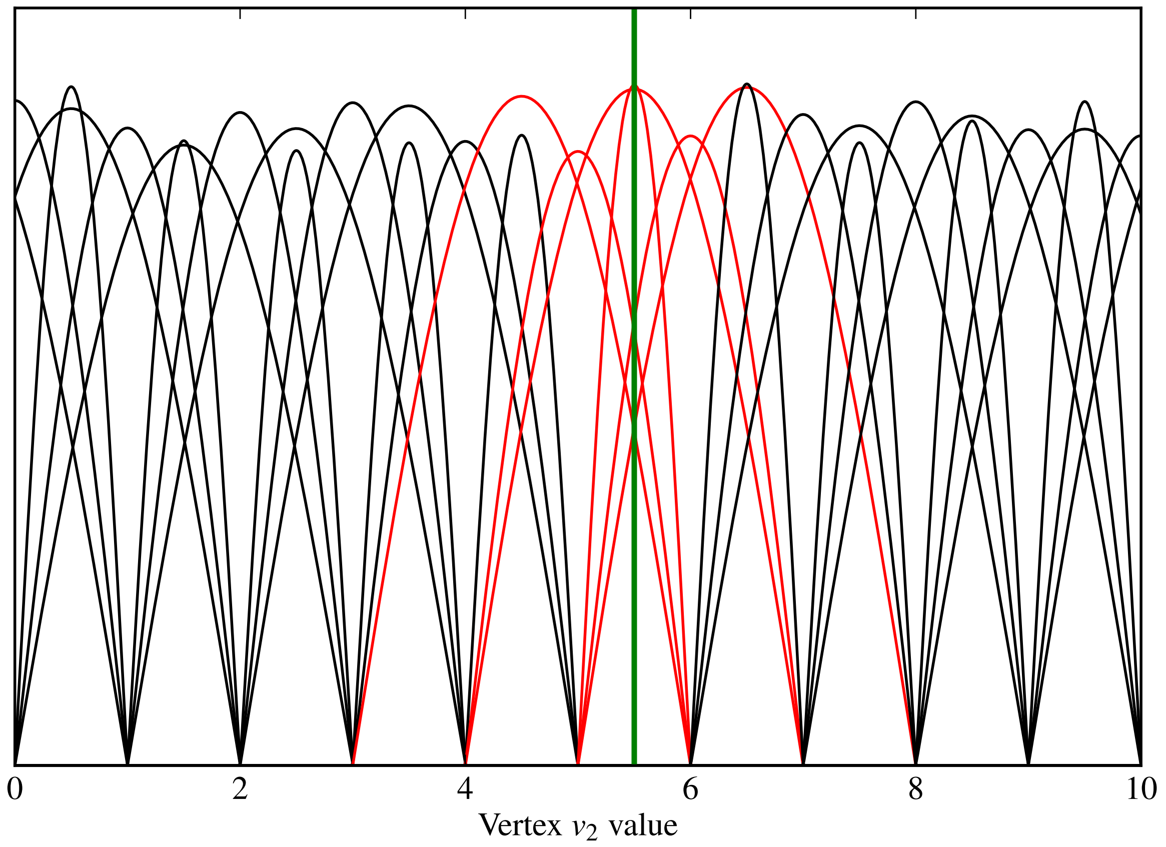

Figure 1: The vertices of $\mathcal{G}$ sorted by v_2 value. Lines represent edges. When λ_2 is close to 1 we expect edges to mostly connect vertices close (in v_2-value) to each other. Thus a cut which minimizes the number of edges from S to $\bar{S}$ (for a fixed size of S) should be of the form $S = \{v|v_2(v) < t\}$. In this figure, S contains only vertices to the left of the green line.

2.2 An Algorithmic Proof

We present a simple algorithm which tries to return the best set of the form presented in 2.1. As we will prove that the 'average' h value over such subsets is less than $\sqrt{2(1-\lambda_2)}$, the returned set must have h less than $\sqrt{2(1-\lambda_2)}$ as well.

Algorithm: Spectral-Partitioning:

1. Input: graph $G = (V, E)$ and vector $x \in \mathbb{R}^n$
2. Sort the vertices of V in non-decreasing order of values in x, that is let $V = \{1, \ldots, n\}$ where $x_1 \leq x_2 \leq \ldots \leq x_n$.
3. Let $i \in \{1, \ldots, n-1\}$ be such that $h(\{1, \ldots, i\})$ is minimum.
4. Output $S = \{1, \ldots, i\}$.

Lemma 1 *Let $G = (V, E)$ be a d-regular graph, $x \in \mathbb{R}^n$ be a vector such that $x \perp 1$ and let M be the normalized adjacency matrix of G. Define*

$$\delta := \frac{\sum_{ij} M_{ij}(x_i - x_j)^2}{\frac{1}{n}\sum_{i,j}(x_i - x_j)^2}$$

and let S be the output of the algorithm on input G and x. Then

$$h(S) \leq \sqrt{2\delta}.$$

Note that if we run the algorithm with the second eigenvector $x = v_2$ then

$$\frac{\sum_{ij} M_{ij}(x_i - x_j)^2}{\frac{1}{n}\sum_{ij}(x_i - x_j)^2} = 1 - \frac{x^T M x}{x^T x} = 1 - \lambda_2.$$

Therefore the above lemma implies the "difficult" direction of Cheeger's Inequality. Note that our algorithm does not necessarily return the optimal cut (call it OPT, that is the cut which minimises h globally). Finding OPT is an NP-hard problem. Putting all this together we will finally have

$$\frac{1-\lambda_2}{2} \leq OPT \leq S \leq \sqrt{2(1-\lambda_2)},$$

where S is the output of the algorithm on input $\mathcal{G}$ and v_2.

2.2.1 Proof of Main Lemma

Our goal is to prove that there exists an i such that $h(\{1, \ldots, i\}) \leq \sqrt{2\delta}$. We will do this by showing that there exists a distribution D over sets S of the form $\{1, \ldots, i\}$ such that

$$\frac{\mathbb{E}_{S\sim D}[\frac{1}{d}E(S, \bar{S})]}{\mathbb{E}_{S\sim D}[\min\{|S|, |\bar{S}|\}]} \leq \sqrt{2\delta}. \tag{1}$$

We need to be careful since

$$\frac{\mathbb{E}_{S\sim D}[\frac{1}{d}E(S, \bar{S})]}{\mathbb{E}_{S\sim D}[\min\{|S|, |\bar{S}|\}]} \neq \mathbb{E}_{S\sim D}\left[\frac{\frac{1}{d}E(S, \bar{S})}{\min\{|S|, |\bar{S}|\}}\right], \tag{2}$$

Moreover, it is certainly not true that the l.h.s. of (2) is a lower bound on the r.h.s. of (2). However, we only need to show the existence of one set S for which

$$\frac{\frac{1}{d}E(S,\bar{S})}{\min\{|S|,|\bar{S}|\}} \leq \sqrt{2\delta}, \tag{3}$$

and this can be done by rearranging (1) and using linearity of expectation to get

$$\mathbb{E}_{S\sim D}\left[\frac{1}{d}E(S,\bar{S}) - \sqrt{2\delta}\min\{|S|,|\bar{S}|\}\right] \leq 0,$$

from which will follow the existence of at least one S satisfying (3), which is all we need. Basically we are using the fact that $\frac{A+B}{C+D} \geq \min\{A/C, B/D\}$. Let us now prove (1). From now on we will assume (with motivation later) that

- $x_{\lfloor n/2 \rfloor} = 0$
- $x_1^2 + x_n^2 = 1$.

These assumptions, while used exclusively to simplify the proof, can be seen as a preprocessing step of the algorithm where first a constant is added such that the first assumption holds and then the vector is scaled so that the second assumption holds.[2] The output of the algorithm does not change under these assumptions as the ordering of elements in x is unchanged. Also note that δ does not change.

We next define the distribution D over the sets of the form $\{1,\dots,i\}$, $i \leq n-1$ as the outcome of the following random process:

- We pick a real value t in the range $[x_1, x_n]$ with probability density function $f(t) = 2|t|$.
- We let $S = \{i : x_i \leq t\}$.

Claim 2 *We have that* $\mathbb{E}_{S\sim D}\min\{S,\bar{S}\} = \sum x_i^2$.

Proof

- The probability that an element $i \leq n/2$ belongs to the smallest set is exactly the probability that t is in the range $[x_i, 0]$. Note that this happens with probability $\int_{x_i}^{0} 2|t|dt = x_i^2$
- The probability that an element $i > n/2$ belongs to the smallest set is exactly the probability that t is in the range $[0, x_i]$ which happens with probability x_i^2.
- The claim now follows from linearity of expectation.

■

Claim 3 *We have that* $\mathbb{E}_{S\sim D}\frac{1}{d}E(S,\bar{S}) \leq \sqrt{2\delta}\sum_i x_i^2$

Note that this claim implies the Lemma as then

$$\frac{\mathbb{E}_{S\sim D}[\frac{1}{d}E(S,\bar{S})]}{\mathbb{E}_{S\sim D}[\min\{|S|,|\bar{S}|\}]} \leq \frac{\sqrt{2\delta}\sum_i x_i^2}{\sum_i x_i^2} \leq \sqrt{2\delta}$$

Proof Note that $\mathbb{E}_{S\sim D}[\frac{1}{d}E(S,\bar{S})] = \frac{1}{2}\sum_{ij} M_{ij}\Pr[(i,j) \text{ is cut by } (S,\bar{S})]$. What is $\Pr[(i,j) \text{ is cut}]$? It is exactly the probability that $t \in [x_i, x_j]$:

[2]This centering and scaling can be done assuming the input vector x is such that $x \perp 1$, but $v_2 \perp 1$ so we're safe. The algorithm would fail if fed the vector v_1, which makes sense as $\lambda_1 = 1$ and we cannot have an upperbound $h(\mathcal{G}) \leq 0$.

- If x_i and x_j are of the same sign this happens with probability $|x_1^2 - x_2^2|$
- If they are of different signs $x_1^2 + x_2^2$
- After some thinking, a good upper bound of both expressions is $|x_i - x_j|(|x_i| + |x_j|)$

Therefore,

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{d}E(S,\bar{S})\right] &\leq \frac{1}{2}\sum_{ij} M_{ij}|x_i - x_j|(|x_i| + |x_j|) \\
&\leq \frac{1}{2}\sqrt{\sum_{ij} M_{ij}(x_i - x_j)^2}\sqrt{\sum_{ij} M_{ij}(|x_i| + |x_j|)^2}
\end{aligned}$$

where we have used the Cauchy-Schwarz Inequality to obtain the second line. We consider the two rooted terms separately. We can use the definition of δ to introduce it into the first term,

$$\begin{aligned}
\sum_{ij} M_{ij}(x_i - x_j)^2 &= \delta\frac{1}{n}\sum_{ij}(x_i - x_j)^2 \\
&= 2\delta\left(\sum_i x_i^2 - \frac{1}{n}\left(\sum_i x_i\right)^2\right) \\
&\leq 2\delta\sum_i x_i^2.
\end{aligned}$$

For the second rooted term, we can upper bound $(|x_i| + |x_j|)^2$ by $2x_i^2 + 2x_j^2$ (since $2a^2 + 2b^2 - (a+b)^2 = (a-b)^2 \geq 0$). This gives us

$$\begin{aligned}
\mathbb{E}\left(\frac{1}{d}E(S,\bar{S})\right) &\leq \frac{1}{2}\sqrt{\sum_{ij} M_{ij}(x_i - x_j)^2}\sqrt{\sum_{ij} M_{ij}(|x_i| + |x_j|)^2} \\
&\leq \frac{1}{2}\sqrt{2\delta\sum_i x_i^2}\sqrt{4\sum_i x_i^2} \\
&= \sqrt{2\delta}\sum_i x_i^2
\end{aligned}$$

where we have used the double-stochasticity of M to obtain the second line above.

■

3 Probabilistically Checkable Proofs(PCP)

In this section we will discuss probabilistically checkable proofs, an idea having huge implications in the development of modern theoretical computer science. Before we jump into PCPs let us review some fundamental concepts in computational complexity theory.

3.1 Preliminaries

Every decidable decision problem such as SAT can be represented as a language L, that is the set of all inputs for which a verifier V gives the answer YES, when given the instance and an appropriate proof as input. For example in the case of SAT the set of all satisfying assignments for a statement forms its language.

NP: A language L is in NP iff there exists a polynomial time verifier V which when given an x and a proof Π of length $poly(|x|)$ verifies whether $x \in L$, with the following properties:

- **completeness:** If $x \in L$ then $\exists$ proof Π such that $V^{\Pi}(x)$ accepts.
- **Soundness:** If $x \notin L$ then $\forall$ proofs Π, the verifier $V^{\Pi}(x)$ rejects.

In the definition of NP the deterministic verifier reads the entire proof and the completeness and soundness hold with probability one. We now introduce the idea of a non-deterministic verifier, which is complete in exactly the same sense as a deterministic verifier but with some probability accepts false proofs.

This line of research was begun in the mid 80s by Goldwasser, Micali, and Rackoff [1, 2] and also independently by Babai [3]. One important result in this area is the PCP theorem [4].

3.2 Probabilistically Checkable Proofs (PCPs)

PCP: A language L is in PCP(r, q) if there exists a probabilistic polynomial time verifier V which, when given an instance x and a proof Π of length $poly(|x|)$, proceeds as follows

1. Reads x and $O(r)$ random bits.
2. Based on x and $O(r)$ random bits, queries $O(q)$ bits from Π.
3. Computes and either accepts or rejects with following properties,
   - **completeness:** If $x \in L$ then $\exists$ proof Π such that $V^{\Pi}(x)$ accepts with probability 1.
   - **Soundness:** If $x \notin L$ the $\forall$ proof Π, the verifier $V^{\Pi}(x)$ accepts with probability at most $\frac{1}{2}$.

Theorem 4 PCP Theorem: *NP = PCP(*$\log(n)$*, 1)*

We will not prove the PCP theorem in class (last year we did but it took 4 lectures), instead we will prove (next week) a weaker result: NP = PCP(poly(n), 1).

Exercise 1 Prove that PCP($\log(n)$, 1) $\subseteq$ NP.

Solution Consider a language L $\in$ PCP($\log(n)$, 1), which by definition has a probabilistic polynomial time verifier. The claim is that from this probabilistic verifier we can construct a deterministic verifier. Given a proof Π and an input instance x, the deterministic verifier runs the probabilistic verifier for all combinations of the $clog(n)$ random bits, a total of $2^{clog(n)} = n^c$ runs, and accepts iff the PCP verifier accepts for every bit strings. Since the probabilistic verifier runs in polynomial time and the number of possible combinations is bounded by n^c the deterministic verifier runs in polynomial time. Completeness follows directly from the completeness of the probabilistic verifier. Soundness follows as for at least half of the bit strings we have a rejection in the probabilistic verifier, and thus a rejection by the deterministic verifier. In fact it suffices to use only $n^c/2 + 1$ of the random bitstrings, as at least one of them will reject.

Exercise 2 Given that GAP 3-SAT is NP-complete, prove that NP $\subseteq$ PCP($\log n, 1$).

GAP 3-SAT instances are 3-SAT instances with the constraint that either they are satisfiable or no more than $1 - \epsilon$ of the clauses can be satisfied simultaneously. Deciding whether a GAP 3-SAT instance is satisfiable is in some sense easier than deciding if a general 3-SAT instance is satisfiable due to the ϵ-gap separating the satisfiable instances from the non-satisfiable ones.

Solution We will prove that GAP 3-SAT $\in$ PCP($\log n$, 1). Then, as any problem in NP can be reduced in polytime to GAP 3-SAT (GAP 3-SAT is NP-complete by assumption), we can construct a probabilistic verifier for any problem in NP: First reduce it to GAP 3-SAT and use the verifier for that problem (which we will now present).

We prove that GAP 3-SAT $\in$ PCP($\log n$, 1). First consider reading $\log n$ random bits, and choosing one of the n clauses accordingly (let the bits read be treated as the binary representation of the clause). Read the 3 assignments in the proof which the clause uses. If the proof is correct, the clause is satisfied (correctness). If the proof is incorrect there is a probability at most $1 - \epsilon$ of accepting, since with probability at least ϵ the randomly selected clause will be not satisfied. If $\epsilon < 0.5$ this is not good enough. To drive this probability down, we read $\lceil \frac{1}{\epsilon} \rceil \log n$ random bits from which we choose $\lceil \frac{1}{\epsilon} \rceil$ clauses. The resulting false acceptance rate is then $(1-\epsilon)^{\lceil \frac{1}{\epsilon} \rceil}$, which is strictly less than $\frac{1}{2}$. Thus GAP 3-SAT is in PCP($\log n$, 1) with constants $C_r = \lceil \frac{1}{\epsilon} \rceil$ and $C_q = 3\lceil \frac{1}{\epsilon} \rceil$.

References

[1] S. Goldwasser, S. Micali and C. Rackoff, The Knowledge Complexity of Interactive Proof Systems, *SIAM J. Comput*(1989), pp. 186–208.

[2] S. Goldwasser, S. Micali and C. Rackoff, The Knowledge Complexity of Interactive Proof Systems, *ACM symposium on Theory of computing*(1985), pp. 291–304.

[3] L Babai, Trading group theory for randomness, *ACM symposium on Theory of computing*(1985), pp. 421-429.

[4] S. Arora and S. Safra, Probabilistic Checking of Proofs: A New Characterization of NP, *Journal of ACM*(1998), pp. 70-122.