CSE 521: Design and Analysis of Algorithms I **Spring 2016**

Lecture 13: Spectral Clustering, Power Method

Lecturer: Shayan Oveis Gharan *May 9th* *Scribe: Nao Hiranuma*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

13.1 Spectral Embedding

Let g be an eigenvector of the normalized Laplacian matrix, $D^{-1/2}LD^{-1/2}$. Then, for $f = D^{-1/2}g$ we have

$$\frac{g^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} g}{g^T g} = \frac{f^T L f}{g^T D^{-1/2} D D^{-1/2} g} = \frac{f^T L f}{f^T D f} = \frac{\sum_{i\sim j}(f_i - f_j)^2}{\sum d(i) f_i^2} = R(f), \tag{13.1}$$

where as usual $i \sim j$ means that i is connected to j in G. Let $g_1, ..., g_K$ be the first k eigenvectors of normalized Laplacian matrix. Define $f^i = D^{-\frac{1}{2}}g^i$. Recall that by variational characterization, $g^1, \dots, g^k$ are minimizers of $\frac{g^T D^{-1/2} L D^{-1/2} g}{g^T g}$. So, by the above equation, $f^1, \dots, f^k$ are minimizers of $R(f)$.

Definition 13.1 (Spectral embedding)**.** *For an integer $k \geq 2$, the spectral embedding of $G(V, E)$ is a mapping $F : V \to \mathbb{R}^{k-1}$ where for each vertex i,*

$$F(i) = (f^2(i), \dots, f^k(i)).$$

Similarly, we can define $R(F)$ as follows:

$$R(F) = \frac{\sum \|F(i) - F(j)\|^2}{\sum_{i\sim j} d_i \|F(i)\|^2} = \frac{\sum_{i\sim j}\sum_{\ell=2}^k (f_i^\ell - f_j^\ell)^2}{\sum_i \sum_{\ell=2}^k d(i) {f_i^\ell}^2}$$

Swapping the order of the sums, we can write

$$R(F) = \frac{\sum_{\ell=2}^k \sum_{i\sim j}(f_i^\ell - f_j^\ell)^2}{\sum_{\ell=2}^k \sum_i d(i) {f_i^\ell}^2} = \frac{\sum_{\ell=2}^k {f^\ell}^T L f^\ell}{\sum_{\ell=2}^k {f^\ell}^T D f} = \frac{\sum_{\ell=2}^k {g^\ell}^T D^{-1/2} L D^{-1/2}}{\sum_{\ell=2}^k {g^\ell}^T g^\ell} = \frac{1}{k}\sum_{i=2}^k \lambda_i,$$

where $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of the normalized Laplacian matrix. The second equality follows by (13.1) and the last equality follows by orthonormality of g^i's. Let the energy of a mapping of a graph G be the sum of the squared of the length of the edges of G,

$$\mathcal{E}(F) = \sum_{i\sim j} \|F(i) - F(j)\|^2 .$$

It can be shown that the spectral embedding has the smallest energy among all *isotropic* embeddings of G into k dimensions. Here, we do not formally define the term *isotropic*, roughly speaking, it corresponds to rotationally invarint embeddings of G.

13.2 Spectral Clustering.

Given a set of data points $X_1, \dots, X_n$, the spectral clustering algorithm of Ng, Jordan and Weiss [NJW02] works as follows:

1) Construct a weighted graph G with vertices $[n]$ and for each pair of vertices i, j let

$$w_{i,j} = \exp(-\left\|X_i - X_j\right\|^2 / \sigma^2)$$

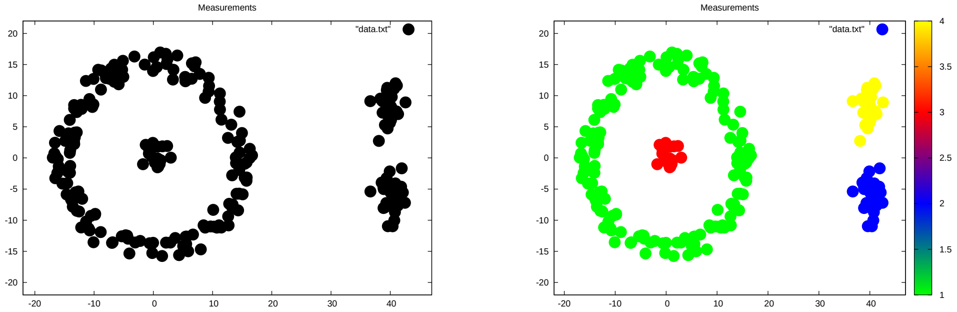
2) for a carefully chosen constant σ.

3) Let $g^1, \ldots, g^k$ be the first k-orthonormal eigenvectors of the normalized Laplacian of G. Compute the spectral embedding of G as defined above. For each pair of vertices i, j let

$$d(i,j) = \left\| \frac{F(i)}{\|F(i)\|} - \frac{F(j)}{\|F(j)\|} \right\|.$$

4) Run the k-means algorithm with respect to the above distance function and return the output as a clustering of the data.

Take a look at the following figure for a simulation of the spectral clustering algorithm. Let us give a high-level intuition of the above algorithm. As we mentioned above, the spectral embedding F is the mapping with minimum energy; because of that it maps the endpoints of edges of G with large weights to close points. Therefore, the vertices who belong to the same cluster will mostly likely map to the same region of $\mathbb{R}^k$. Consequently, even though the k-means algorithm was unable to find the clusters with respect to the original data points, it can correctly find them with respect to the spectral embedding distance function.



Ng, Jordan and Weiss do not provide any theory performance guarantee of the spectral clustering algorithm. Recently, there has a been major progress on proving a theoretical bound on the quality of the output of this algorithm [LOT12; Lou+12; PSZ15].

Theorem 13.2 ([LOT12])**.** *For a given graph $G = (V, E)$ and an integer $k \geq 2$, let*

$$\phi_k(G) = \min_{\text{disjoint } S_1, S_2, \ldots, S_k} \max_{S_i} \phi(S_i),$$

be the maximum conductance of the best k disjoint clusters of G. Then,

$$\lambda_k \leq \phi_k(G) \leq O(k^2 \cdot \sqrt{\lambda_k}),$$

where λ_k is the k-th smallest eigenvalue of the normalized Laplacian of G. Furthermore, there is an algorithm which returns k disjoint sets $S_1, \ldots, S_k$ such that $\max_i \phi(S_i) \leq O(k^2 \sqrt{\lambda_k})$.

13.3 Power method

As alluded to in the previous lectures, SVD is a very time consuming algorithm. It typically takes $O(n^3)$ to run SVD on an $n \times n$ matrix. In this section we discuss a fast algorithm known as the Power method to approximately find the largest (or a few largest) eigenvalues of a given PSD matrix.

Algorithm 1 Power method

```
Let $x \in R^n$ be a Gaussian vector, i.e., each coordinate of $x$ is an independent standard normal random variable. Set $x_0 \leftarrow x$.
for $i = 1 \to k$ do
    $x_i = M x_{i-1}$
end for
return $\frac{x_k^T M x_k^T}{x_k^T x_k^T}$.
```

Note that k is a parameter of the algorithm. For larger values of k the algorithm gives better approximation of the largest eigenvalue of M as stated in the following theorem.

Theorem 13.3. *For any PSD matrix $M \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_1 \geq, ..., \geq \lambda_n \geq 0$, and any integer $k > 0$ and $\epsilon > 0$, with constant probability,*

$$\frac{x_k^T M x_k}{x_k^T x_k} \geq \frac{\lambda_1(1-\epsilon)}{1+10n(1-\epsilon)^{2k}}$$

Therefore, to get a $1-\epsilon$ multiplicative approximation of the largest eigenvalue of M it is enough to let $k = 2\frac{\log n}{\epsilon} \log \frac{1}{\epsilon}$, so that

$$(1-\epsilon)^{2k} \approx e^{-4\log(n)\log\frac{1}{\epsilon}} \sim \frac{\epsilon}{50n},$$

and with constant probability we get

$$\frac{x_k^T M x_k}{x_k^T x_k} \geq \frac{\lambda_1(1-\epsilon)}{1+\epsilon/5} \geq \lambda_1(1-2\epsilon).$$

In addition observe that the algorithm only uses k matrix vector product. Each matrix vector product can be computed in time $O(\text{nnz}(M))$ where $\text{nnz}(M)$ is the number of nonzero entries of M, i.e., it is the length of the input to the algorithm. Therefore, Algorithm 1 in time $O(\log(n)\,\text{nnz}(M)\log\epsilon^{-1}/\epsilon)$ returns a $1-\epsilon$ multiplicative approximation of the largest eigenvalue of M with a constant probability.

In the rest of this section we prove Theorem 13.3. We break the proof to 3 lemmas. Throughout the proof assume that $v_1, \ldots, v_n$ are orthonormal eigenvectors of M corresponding to $\lambda_1 \geq \lambda_2 \geq \ldots$
$geq\lambda_n$.

Lemma 13.4. $\mathbb{P}\left[|\langle x, v_1\rangle| \geq \frac{1}{2}\right] \geq \Omega(1)$

This lemma naturally follows from the rotational invariance property Gaussian vectors. In particular $\langle x, v_1$ is distributed as a standard normal random variable; so with a constant probability $|\langle x, v_1\rangle|$ is more than 1/2.

Lemma 13.5. $\mathbb{P}\left[\|x\|^2 \leq 2n\right] \geq 1 - e^{-\frac{n}{8}}$.

The proof of the above lemma follows from the concentration inequality of the sum of squares of independent standard normal random variables that we discussed in Lecture 6.

Lemma 13.6. *For all any vector $x \in \mathbb{R}^n$, and $y = M^k x$ we have*

$$\frac{y^T M y}{y^T y} \geq \frac{(1-\epsilon)\lambda_1}{1 + \frac{\|x\|^2}{\langle x, v_1\rangle^2}(1-\epsilon)^{2k-1}}$$

Note that the above statement holds for any vector x, and we not going to use the randomness of x anymore. It is easy to see that Theorem 13.3 follows from the above 3 lemmas. In particular by lemmas 13.4 and 13.5 with a constant probability $\|x\|^2 / \langle x, v_1\rangle^2 \leq 4n$, which implies the theorem.

Proof. Let

$$x = \sum_{i=1}^n a_i v_i,$$

where for each i, $a_i = \langle x, v_i\rangle$. First, observe that

$$\begin{aligned} y^T M y &= x^T M^{2k+1} x \\ &= \left(\sum_{i=1}^n a_i v_i\right) M^{2k+1} \left(\sum_{i=1}^n a_i v_i\right) \\ &= \left(\sum_{i=1}^n a_i v_i\right)\left(\sum_{i=1}^n a_i \lambda_i^{2k+1} v_i\right) = \sum_{i=1}^n a_i^2 \lambda_i^{2k+1}, \end{aligned}$$

where the last equality follows by orthonormality of $v_1, \ldots, v_n$. Similarly, we can write

$$y^T y = x^T M^{2k} x = \sum_{i=1}^n a_i^2 \lambda_i^{2k}$$

Therefore, to prove the lemma it is enough to show that

$$\frac{\sum_{i=1}^n a_i^2 \lambda_i^{2k+1}}{\sum_{i=1}^n a_i^2 \lambda_i^{2k}} \geq \frac{(1-\epsilon)\lambda_1}{1 + \frac{\|x\|^2}{a_1}(1-\epsilon)^{2k-1}}. \tag{13.2}$$

The rest of the proof is an algebraic manipulation of the ratio on the LHS.

Choose an integer $j \geq 1$ such that $\lambda_j \geq (1-\epsilon)\lambda_1$ and $\lambda_{j+1} < (1-\epsilon)\lambda_1$. Think of $\lambda_1, \ldots, \lambda_j$ as "big" eigenvalues and the rest of them as "small" eigenvalues. The main intuition of is that the small eigenvalues have negligible contribution to the numerator and denominator so we we can throw them away. On the other hand, for any big eigenvalue λ_i the ratio of the contributions of λ_i is at least $\frac{a_i \lambda_i^{2k+1}}{a_i \lambda_i^{2k}} = \lambda_i \geq (1-\epsilon)\lambda_1$.

First, observe that

$$\sum_{i=1}^n a_i \lambda_i^{2k+1} \geq \lambda_1(1-\epsilon) \sum_{i=1}^j a_i^2 \lambda_i^{2k} \tag{13.3}$$

Secondly, observe that

$$\sum_{i=j+1}^n a_i^2 \lambda_i^{2k} \leq \sum_{i=j+1}^n a_i^2 \lambda_1^{2k}(1-\epsilon)^{2k} = (1-\epsilon)^{2k} \lambda_1^{2k} \sum_{i=j+1}^n a_i^2 \leq \|x\|^2 \lambda_1^{2k}(1-\epsilon)^{2k}$$

So,

$$\begin{aligned}\frac{y^T M y}{y^T y} &\geq \frac{\lambda_1(1-\epsilon)\sum_{i=1}^{j} a_i^2 \lambda_i^{2k}}{\sum_{i=1}^{j} a_i^2\lambda_i^{2k} + \|x\|^2 \lambda_1^{2k}(1-\epsilon)^{2k}} \\ &\geq \frac{1}{\frac{\sum_{i=1}^{j} a_i^2\lambda_i^{2k}}{\lambda_1(1-\epsilon)\sum a_i^2\lambda_i^{2k}} + \frac{\|x\|^2\lambda_1^{2k}(1-\epsilon)^{2k}}{a_1^2\lambda_1^{2k}(1-\epsilon)\lambda_1}} \\ &= \frac{1}{\frac{1}{\lambda_1(1-\epsilon)} + \frac{\|x\|^2(1-\epsilon)^{2k-1}}{a_1^2}} \geq \frac{(1-\epsilon)\lambda_1}{1+\frac{\|x\|^2}{a_1^2}(1-\epsilon)^{2k-1}}\end{aligned}$$

This proves (13.2) which completes the proof of Lemma 13.6 and Theorem 13.3. □

We can use an algorithm similar to Algorithm 1 to estimate the second largest eigenvector of M. Suppose we know the largest eigenvector v_1 of M and let x be a random Gaussian vector. Then, we let

$$x_0 = x - \langle v_1, x\rangle v_1,$$

i.e., we project x onto the space orthogonal to v_1. The rest of the algorithm works as before, for each $i \geq 1$ we let

$$x_i = M x_{i-1},$$

and we return $\frac{x_k^T M x_k}{x_k^T x_k}$. The same proof shows that with a constant probability

$$\frac{x_k^T M x_k}{x_k^T x_k} \geq \frac{\lambda_2(1-\epsilon)}{1+10n(1-\epsilon)^2 k}.$$

Now, let's see if we can use the same idea to approximate the 2nd smallest eigenvector of the normalized Laplacian matrix. Firstly, note that the performance of the spectral partitioning algorithm is robust to the Rayleigh quotient. That is, for any given vector y such that $y \perp 1$, the algorithm returns a set S such that

$$\phi(S) \leq O\left(\sqrt{\frac{y^T \frac{L}{d} y}{y^T y}}\right).$$

On idea is to use the PSD matrix $M = 2I - \frac{L}{d}$. Observe that the 2nd largest eigenvector of M is the same as the 2nd smallest eigenvector of L/d. So, by Theorem 13.3, with constant probability, Algorithm 1 returns a vector y such that

$$y^T M y \geq (1-\epsilon)(2-\lambda_2) y^T y$$

where λ_2 is the 2nd smallest eigenvalue of L/d. Using the definition of M,

$$y^T M y = y^T(2I - L/d)y = 2y^T y - y^T \frac{L}{d} y$$

Therefore,

$$\frac{y^T \frac{L}{d} y}{y^T y} \leq \lambda_2 + 2\epsilon$$

So, to get a constant factor approximation of λ_2 we have to let $\epsilon = O(\lambda_2)$. But in that case, the running time of the power method is $O(\log(n)/\lambda_2)$. So, for a cycle it runs in time $O(n^2 \log n)$ which is very slow.

Instead, the idea is to use the inverse of the normalized Laplacian matrix. Unfortunately, we are not aware of any linear time algorithms to compute the inverse of a matrix. But, to run the power method it is enough to solve systems of linear of equations. In particular, instead of letting $x_i = (L/d)^{-1} x_{i-1}$, we can solves the following systems of linear equations

$$\frac{L}{d} x_i = x_{i-1}.$$

There are very fast algorithms that solve a system of linear equations corresponding to a Laplacian matrix of a graph. We do not cover these algorithms in this course but we refer interested students to [ST04; KMP10; KMP11; Kel+13; KS16].

References

[Kel+13] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time". In: *STOC*. 2013, pp. 911–920 (cit. on p. 13-6).

[KMP10] I. Koutis, G. L. Miller, and R. Peng. "Approaching Optimality for Solving SDD Linear Systems". In: *FOCS*. 2010, pp. 235–244 (cit. on p. 13-6).

[KMP11] I. Koutis, G. L. Miller, and R. Peng. "A Nearly-m log n Time Solver for SDD Linear Systems". In: *FOCS*. 2011, pp. 590–598 (cit. on p. 13-6).

[KS16] R. Kyng and S. Sachdeva. "Approximate Gaussian Elimination for Laplacians: Fast, Sparse, and Simple". 2016. URL: http://arxiv.org/abs/1605.02353 (cit. on p. 13-6).

[LOT12] J. R. Lee, S. Oveis Gharan, and L. Trevisan. "Multi-way spectral partitioning and higher-order cheeger inequalities". In: *STOC*. 2012, pp. 1117–1130 (cit. on p. 13-2).

[Lou+12] A. Louis, P. Raghavendra, P. Tetali, and S. Vempala. "Many sparse cuts via higher eigenvalues". In: *STOC*. 2012 (cit. on p. 13-2).

[NJW02] A. Ng, M. Jordan, and Y. Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *NIPS*. 2002 (cit. on p. 13-1).

[PSZ15] R. Peng, H. Sun, and L. Zanetti. "Partitioning Well-Clustered Graphs with k-Means and Heat Kernel". In: *COLT*. 2015, pp. 1423–1455 (cit. on p. 13-2).

[ST04] D. A. Spielman and S.-H. Teng. "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *STOC*. 2004, pp. 81–90 (cit. on p. 13-6).