CPS290: Algorithmic Foundations of Data Science **March 7, 2017**

Lecture 12 : Graph Laplacians and Cheeger's Inequality

Lecturer: Kamesh Munagala *Scribe: Kamesh Munagala*

Graph Laplacian

Maybe the most beautiful connection between a discrete object (graph) and a continuous object (eigenvalue) comes from a concept called the *Laplacian*. It gives a surprisingly simple and algebraic method for splitting a graph into two pieces without cutting too many edges. In a sense, we take an NP-Complete problem of partitioning a graph and relax it to an easy to solve problem of finding an eigenvector of a certain type of matrix. The resulting partition is not optimal, but close to it in a certain provable sense. The algorithm is very practical and modifications of it are widely used for *clustering* objects into similar groups.

Suppose we are given n items and some measure of similarity between them. Let w_{ij} denote the similarity score between items i and j. Let us assume this function is symmetric, so that $w_{ij} = w_{ji}$. We will treat the general version later (without proofs), but will consider a simpler version for presenting proofs. Let us assume $w_{ij} \in \{0, 1\}$, where $w_{ij} = 1$ means i and j are similar, and dissimilar otherwise. This defines an undirected, unweighted graph $G(V, E)$, where V is the set of items, and E is the set of edges corresponding to pairs of items that are similar. We further assume all vertices in this graph have degree exactly d. We will relax all these assumptions later.

Let A denote the adjacency matrix of G, and let D denote the diagonal matrix whose diagonal entries are all d. Then, the *unnormalized Laplacian* of G is given by

$$\mathcal{L} = D - A$$

Basic Properties

It is easy to check that for any vector $\vec{v}$, we have

$$\vec{v}^T \mathcal{L} \vec{v} = \sum_{(i,j) \in E, i<j} (v_i - v_j)^2$$

Since the above quantity is always non-negative, the matrix $\mathcal{L}$ is PSD, and has eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Further, note that if $\vec{v} = \vec{1}$, then $\vec{v}^T \mathcal{L} \vec{v} = 0$. This means $\lambda_1 = 0$ and a corresponding eigenvector is $\vec{1}$. We now show something stronger.

Claim 1. *Suppose $0 = \lambda_1 = \cdots = \lambda_k < \lambda_{k+1}$, then the number of connected components is exactly k.*

Proof. We will show that there are exactly k orthonormal eigenvectors for the 0 eigenvalue, when the graph has k connected components. First note that if $\vec{v}$ is a unit length eigenvector corresponding to any eigenvalue λ, then

$$\vec{v}^T \mathcal{L} \vec{v} = \sum_{(i,j) \in E, i<j} (v_i - v_j)^2 = \lambda$$

Suppose $\lambda = 0$, then it is easy to check that in the corresponding eigenvector $\vec{v}$, for any connected component S_i, all vertices $j \in S_i$ have the same value v_j. If there are k connected component, we can construct k orthogonal eigenvectors as follows: For each $i = 1, 2, \ldots, k$, construct vector $\vec{v_i}$ by setting $v_{ij} = 1$ for $j \in S_i$ and zero everywhere else. These k vectors are clearly orthogonal, which shows that the multiplicity of the zero eigenvalue is at least k.

On the other hand, there is no other eigenvector for the zero eigenvalue that is orthogonal to these k. To see this, any such eigenvector must take the same value on all vertices within a connected component. So it's dot product with any of the k eigenvectors we constructed above has to be non-zero. This shows that the multiplicity of the zero eigenvalue is exactly the number of components k, that is, the corresponding eigenvectors span a subspace of dimension k. □

Graph Partitioning

If we only wish to find the number of connected components, the Laplacian seems like a convoluted way to go about doing it. In practice, typically the graph is connected, and we need to split it into components such that items within a component are much more similar with each other than with items in a different component. Such a problem is usually termed *clustering* or *unsupervised learning*. We will now consider the simplest problem of splitting the graph into two pieces. We want the partition to have two properties: The number of edges going from one side of the partition to the other is small; and the number of edges within each partition is large. One way to do it is to find a minimum cut in the graph – a partition that minimizes the number of edges going across. This can be done by network flows in polynomial time (more later), but simply minimizing the number of edges crossing the partition can lead to silly solutions, for instance, it may be optimal to split off one vertex from everything else.

A more reasonable solution would attempt to minimize the number of edges while simultaneously making sure each side of the partition has a lot of edges. Let $(S, V \setminus S)$ be a partition. We define the cut $\delta(S)$ as the set of edges with one end-point in S and the other in $V \setminus S$. Similarly, we define the volume of the partition as $\text{vol}(S) = \sum_{i \in S} d_i$, where d_i is the degree of i. Since the graph is regular, this is the same as $d|S|$. Given these, we can define a good partition as one that minimizes the *conductance*, defined as:

$$\frac{|\delta(S)|}{\min\left(\text{vol}(S), \text{vol}(V \setminus S)\right)} = \frac{1}{d} \frac{|\delta(S)|}{\min\left(|S|, |V \setminus S|\right)}$$

This attempts to simultaneously decrease the cut size while increase the volume of the partitions. Without loss of generality, we can define S to be the smaller side of the partition, and we can ignore the factor of d. Therefore, define

$$\Theta_G = \min_{S \subseteq V, |S| \leq |V|/2} \frac{|\delta(S)|}{|S|}$$

Our goal is to find an S that minimizes the RHS. This is the *minimum normalized cut* problem. Note that $\frac{\Theta_G}{d}$ is the *conductance* of the graph defined above. The quantity $\frac{|\delta(S)|}{|S|}$ is termed the *normalized cut size* of cut S; for a regular graph, conductance is proportional to the normalized cut size, so minimizing either of them is the same.

Connection to Laplacian

Finding a cut of minimum conductance is NP-Complete. However, conductance is closely related to the graph Laplacian. Take any set $S \subset V$ of size at most $|V|/2$. Consider the vector $\vec{v_S}$ defined

as follows: For $j \in S$, $v_{Sj} = 1 - |S|/|V|$, and for $j \notin S$, we set $v_{Sj} = -|S|/|V|$. It is an easy exercise to see that $\vec{v_S}$ is perpendicular to $\vec{1}$. Further,

$$\langle \vec{v_S}, \vec{v_S} \rangle = |S| \left(1 - \frac{|S|}{|V|}\right)^2 + (|V| - |S|) \left(\frac{|S|}{|V|}\right)^2 = \frac{|S||V\S|}{V|}$$

and

$$\vec{v_S}^T \mathcal{L} \vec{v_S} = |\delta(S)| \left(1 - |S|/|V| + |S|/|V|\right)^2 = |\delta(S)|$$

Therefore,

$$\frac{\vec{v_S}^T \mathcal{L} \vec{v_S}}{\vec{v_S}^T \vec{v_S}} = \frac{|\delta(S)|}{\frac{|S||V\setminus S|}{|V|}} = \frac{|\delta(S)|}{|S|} \times \frac{1}{1 - |S|/|V|} = c\frac{|\delta(S)|}{|S|}$$

where $c \in [1, 2]$ since $|S| \leq |V|/2$. The LHS of the above relation is at least λ_2, since

$$\lambda_2 = \min_{\vec{v} \perp \vec{1}} \frac{\vec{v}^T \mathcal{L} \vec{v}}{\vec{v}^T \vec{v}}$$

and since $\vec{v_S} \perp \vec{1}$. Since all this holds for any S, it holds for the infimum, so that

$$\lambda_2 \leq 2 \min_{S s.t. |S| \leq |V|/2} \frac{|\delta(S)|}{|S|} = 2\Theta_G$$

Therefore, the minimum normalized cut size of a regular graph is at least $\lambda_2/2$. In fact, if we restricted to unit vectors with two entries, one positive and one negative, that are perpendicular to $\vec{1}$, then the quantity $\vec{v^T} \mathcal{L} \vec{v}$ is (within a factor of 2) the normalized cut size of the cut that separates the vertices where $\vec{v}$ takes the positive value from the vertices that $\vec{v}$ takes negative values. The minimum conductance cut therefore (roughly speaking) minimizes $\vec{v^T} \mathcal{L} \vec{v}$ over unit vectors perpendicular to $\vec{1}$ that takes only two values for all its coordinates. However, optimizing over this set of vectors is NP-Complete. What we can do instead is optimize over the space of all unit vectors $\vec{v}$ that are perpendicular to $\vec{1}$. This yields the second eigenvector of $\mathcal{L}$ as its optimal solution. Clearly, the resulting objective value, the second eigenvalue, is smaller than the normalized cut size. This is called a *relaxation* of the original problem, since we replaced a hard discrete optimization problem (conductance) with an easier continuous optimization problem (eigenvalue).

The Spectral Method

The question then becomes: Suppose we solve for the second eigenvector. Can we convert this to a good cut? If so, how good is this cut? This is the *spectral algorithm*.

- Let $\vec{v}$ denote the second smallest eigenvector of $\mathcal{L}$. Sort the vertices i of G in increasing order of v_i. Let the resulting ordering be $v_1 \leq v_2 \leq \cdots \leq v_n$.
- For each i, consider the cut $\{1, 2, \ldots, i\}$ and its complement. Calculate its conductance.
- Among these $n - 1$ cuts, choose the one with minimum conductance.

The above algorithm need not find the optimum conductance cut, but has the advantage of being efficient – all it involves is an eigenvalue computation, followed by sorting, followed by calculating the conductance of $n-1$ cuts. How good is this algorithm? This is shown by the following theorem.

Theorem 1 (Cheeger's Inequality)**.** *Let G be a regular connected graph with conductance $\frac{\Theta_G}{d}$. Let Γ denote the normalized cut size of the cut found by the spectral algorithm, and let λ_2 denote the value of the second smallest eigenvector of the Laplacian. Then*

$$\frac{\lambda_2}{2} \leq \Theta_G \leq \Gamma \leq \sqrt{2d\lambda_2}$$

This shows that the spectral method has pretty good performance – its conductance is roughly the square root of the optimal conductance. So if the optimum is very small, the spectral method will output a decent cut. This gives an extremely practical way to partition a graph into two pieces so that the vertices within a piece are on average more similar to each other than to vertices in the opposite side. The proof of this theorem is not hard, but beyond the scope of this course.

General Graph

In the above discussion, we assumed the graph is regular. Even if the graph is arbitrary, the Laplacian can be constructed in the same way, and the spectral algorithm can be run as before. However, Cheeger's inequality only holds for a slightly different definition of Laplacian, termed the *Normalized Laplacian*.

Define the volume of a set S as $\text{vol}(S) = \sum_{i \in S} d_i$. Define the conductance of the graph as:

$$\phi_G = \min_{S | \text{vol}(S) \leq \text{vol}(V)/2} \frac{|\delta(S)|}{\text{vol}(S)}$$

Note that for a regular graph, $d\phi_G = \Theta_G$.

Next define the normalized Laplacian as $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$, where A is the adjacency matrix, and D is the diagonal matrix of degrees. As before, compute the second eigenvector of $\mathcal{L}$ and apply the spectral method to construct $n-1$ cuts and choose the minimum conductance cut among them. Then

Theorem 2 (Cheeger's Inequality)**.** *Let G be a connected graph with conductance ϕ_G. Let γ denote the conductance of the cut found by the spectral algorithm applied to the normalized Laplacian, and let λ_2 denote the value of the second smallest eigenvector of the normalized Laplacian. Then we have the following result.*

$$\frac{\lambda_2}{2} \leq \phi_G \leq \gamma \leq \sqrt{2\lambda_2}$$

Note that for regular graphs, both theorems are identical – the normalized Laplacian is $1/d$ times the Laplacian, which means the second eigenvalue is $1/d$ times smaller. Further, the normalized cut size is d times the conductance. If you put these together, you can show that both theorems are saying the same thing. The main point is that for non-regular graph, the "right" Laplacian is the normalized one, and the "right" measure of the quality of a cut is its conductance. However, we can use the Laplacian $D - A$ for running the algorithm; in practice, it does reasonably well.

We can further generalize the algorithm to handle weights on edges. The generalization is simple if we replace degree with weighted degree, and volume by weighted volume. The definitions above generalize naturally, and so does the algorithm, and performance guarantee.

Generalization to Many Partitions

To partition a graph into more pieces, one approach is to consider higher eigenvectors. For instance, we can take the smallest k eigenvectors, and project the vertices onto this space. This given n points

in k dimensional Euclidean space. These points are usually simpler to partition into k groups, since the eigenvectors have already done most of the hard work! Suppose we want to partition the resulting points into k groups, then one simple heuristic is the following, called *k-center*:

- Map each vertex to its projection onto the first k eigenvectors of the Laplacian (normalized or unnormalized).
- Start with any vertex; place it in partition 1.
- Take the vertex furthest away from it and place it in partition 2.
- Then take the vertex whose minimum distance to the first two vertices is largest and place it in partition 3, and so on till we have populated k partitions with one vertex each.
- Call these vertices, one per partition, the *centers* of the partitions.
- For the remaining vertices, place each of them in that partition whose center is closest to it.

This gives a partition of the n vertices of a graph into k groups. In practice, such schemes work extremely well and is called the *spectral clustering* algorithm. For $k = 2$, this essentially takes the second eigenvector, makes the two extreme vertices as centers of the partitions, and places each vertex in that partition whose center is closer to it. This heuristic is worse than the algorithm that achieves Cheeger's inequality, but is simpler than computing the conductance of $n - 1$ cuts.