Graphs and Networks Lecture 6

Spectral Partitioning

Lecturer: Daniel A. Spielman September 25, 2007

6.1 Cut Problems

Graph partitioning problems typically reqire one to divide the vertices of a graph into two or more large pieces, while cutting as few edges as possible. If the graph is weighted, this probably means minimizing the sum of the weights of the cut edges. There are many possible constraints that could be put on the sizes of the pieces produced. For example, the graph bisection problem is to find a set $S \subset V$ such that $|S| = \lfloor |V|/2 \rfloor$ minimizing

$$|\{(i,j) \in E : i \in S, j \not\in S\}|.$$

We will refer to a set $S \subseteq V$ as a *cut*, by which we mean the partition of V into S and its complement.

But, a strict bisection is often not exactly what one wants. There are various ways of loosening the bisection constraint. A common approach is to ask for a α-balanced cut, which requires that $\alpha n \leq |S| \leq (1-\alpha)n$. In today's lecture, we will consider measures of cut quality that compare the number of edges cut to the number of vertices or edges removed.

We define the *sparsity* of a cut S to be

$$\mathrm{sp}(S) \stackrel{\mathrm{def}}{=} \frac{|\partial(S)|}{\min(|S|, |V-S|)},$$

where $\partial(S)$ is the boundary of S: all edges with one endpoint in S and the other outside of S.

This measure weights a cut by the number of vertices inside, and values higher cuts that are more balanced. The *sparsest cut problem* is to find the set S minimizing $\mathrm{sp}(S)$.

It is NP-hard to find the *sparsest cut* in a graph, and all the variations we will consider will be NP-hard as well. So, we will satisfy ourselves by finding approximations of the sparsest cut. Today, we will see that these can be found by computing eigenvectors. (The computational hardness of approximating these problems is not yet well understood).

The problems of finding sparse cuts in graphs (and its variations) have proved useful in many applications. Some that come to mind are:

1. The division of a computational problem among parallel processors, where the vertices represent parts of the problem, and the edges represent communications.
2. The clustering of graph-based data.
3. The design of divide-and-conquer algorithms.

6.2 Examples

Here are some examples of graphs, along with their sparsest cuts.

1. In the path graph on n vertices, with n even, the sparsest cut is the first set of left-most $n/2$ vertices.
2. In the $\sqrt{n}$-by-$\sqrt{n}$ grid, the sparsest cut is also the set of left-most $n/2$ vertices, provided that $\sqrt{n}$ is even.
3. If we add a tail of length $k > 2\sqrt{n}$ to the above grid (to make a kite), then the sparsest cut is just the tail

6.3 Variations

A variation of the sparsity of a cut that we will typically find more convenient to work with will be called the *ratio* of a cut, defined by

$$r(S) \stackrel{\text{def}}{=} \frac{|\partial(S)|}{|S|\,|V-S|}.$$

This was considered in the works (Leighton-Rao 1988, Hagen-Kahng 1992, Wei-Cheng 1989). The ratio of a cut has some advantages over the sparsity of a cut. The main one is that it doesn't have a minimum in it, which makes it a little closer to being continuous. That said, they are very close. If $|S| \leq n/2$, then $|V-S| \geq n/2$. So, we have

$$\frac{\mathrm{sp}(S)}{n} \leq r(S) \leq \frac{\mathrm{sp}(S)}{n/2}.$$

The ratio of a cut is also closer to the conductance of a cut, a quantity that we will see in a moment, that has been useful in partitioning problems.

In the definition of conductance, we weight vertices by their degrees. That is, we consider how many edges are attached to each side of the graph, rather than just the number of vertices. For a set of vertices S, set

$$d(S) \stackrel{\text{def}}{=} \sum_{u \in S} d(u),$$

where $d(u)$ is the degree of vertex u. Then, the conductance of a set S is given by

$$\phi(S) \stackrel{\text{def}}{=} \frac{|\partial(S)|}{d(S)d(V-S)}.$$

(Note that some use $\min(d(S), d(V-S))$ in the denominator instead).

Finally, we will also define the conductance of weighted graphs. The definition is the natural one. In a weighted graph, we define the weighted degree of a vertex to be the sum of the weights of the edges attached to it:

$$d(u) = \sum_{v:(u,v)\in E} w(u,v).$$

Then, the conductance of a set S is given by

$$\phi(S) \stackrel{\text{def}}{=} \frac{w(\partial(S))}{d(S)d(V-S)}.$$

We can then consider the problem of finding the set S of minimum conductance. In particular, we define the conductance of a graph G by

$$\Phi(G) \stackrel{\text{def}}{=} \min_{S \subset V} \phi(S).$$

6.4 Obstacles to Rapid Mixing

A set of low-conductance is an obstacle to rapid mixing in a graph. To see this, let S be a set, and let $\boldsymbol{p}_0$ be the initial distribution that we obtain by restricting the stable distribution to S:

$$\boldsymbol{p}_0(u) = \begin{cases} d(u)/d(S) & \text{if } u \in S \\ 0 & \text{otherwise.} \end{cases}$$

I will now convince you that if one starts at this distribution, then after $d(V)/4\phi(S)$ steps, the random walk will be far from mixed. It is also possible to make a formal proof of this, but I won't have time in class. The key point of our proof will be to show that after t steps, the probability of being outside of S will be at most $t\phi(S)d(V)$.

Let's look at what the (ordinary not lazy) random walk does in the first time step. A node $u \in S$ has probability mass $d(u)/d(S)$, and so it sends probability mass $1/d(S)$ to each of its neighbors. The number of edges leaving S is $\phi(S)d(S)d(V-S)$, so the total probability mass leaving S is $\phi(S)d(V-S)$. One can in fact prove that at most this much mass leaves S in every successive step. So, after t steps, the chance of being outside S is at most

$$t\phi(S)d(V-S) \leq t\phi(S)d(V).$$

So, when $\phi(S)$ is small, the random walk will take a long time to mix. (because if $d(S) < d(V)/2$, then for the walk to mix half the mass should be outside S.)

6.5 Relation to Eigenvectors and Eigenvalues

We will now show a relation between the ratio cut problem and eigenvectors and eigenvalues. In the next section, we will develop this relation for conductance.

Eventually, we will state Cheeger's inequality, which says that

$$\frac{\lambda_2}{d(V)} \leq \Phi(G) \leq \frac{2\sqrt{2\lambda_2}}{d(V)}.$$

The value λ_2 in this expression is the same one from two lectures ago. That is $\lambda_2 = 1 - \mu_2$, where μ_2 was the second-largest eigenvalue of the walk matrix.

We will see how to find approximate sparset cuts by using one of the greatest ideas in optimization: relaxation. The idea of relaxation is to take a problem that is fundamentally discrete, and turn it into a continuous problem. Here, our problem requires us to find a set S. To turn the problem into a continuous one, we need to formulate it in an arithmetic fashion. To begin, we consider vectors instead of sets. In particular, we will represent S by the vector χ_S, defined by

$$\chi_S(u) = \begin{cases} 1 & \text{for } u \in S \\ 0 & \text{for } u \notin S. \end{cases}$$

But, we will write a formula that makes sense for any $\boldsymbol{x} \in \mathbb{R}^V$. We need a formula for

$$|\{(i,j) \in E : i \in S, j \notin S\}|.$$

We will use

$$\sum_{(u,v)\in E} (\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.$$

Similarly, for $|S|\,|V - S|$, we will use

$$\sum_{u \neq v} (1/2)(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.$$

So, our problem becomes

$$\min_{\boldsymbol{x}\in\{0,1\}^n} \frac{\sum_{(u,v)\in E}(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.}{\sum_{u\neq v}(1/2)(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.}.$$

To relax this problem, we will allow $\boldsymbol{x}$ to range over all of $\mathbb{R}^n$. This will give a solution of lower value, but it probably won't be 0/1 valued. When we allow $\boldsymbol{x}$ to be real-valued, we see that some simplification is possible: if we added a constant to each entry of $\boldsymbol{x}$, then the ratio wouldn't change. So, we will normalize by assuming $\sum_u \boldsymbol{x}(u) = 0$. In this case, we also have that

$$\sum_{u\neq v}(1/2)(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2. = n\sum_u \boldsymbol{x}(u)^2.$$

So, our relaxation becomes

$$\begin{aligned}
\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{\sum_{(u,v)\in E}(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.}{\sum_{u\neq v}(1/2)(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.} &= \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{x}\perp\mathbf{1}} \frac{\sum_{(u,v)\in E}(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.}{\sum_{u\neq v}(1/2)(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.} \\
&= \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{x}\perp\mathbf{1}} \frac{\sum_{(u,v)\in E}(\boldsymbol{x}(u) - \boldsymbol{x}(v))^2.}{n\sum_u \boldsymbol{x}(u)^2.} \\
&= \frac{1}{n} \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{x}\perp\mathbf{1}} \frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}},
\end{aligned}$$

where $\boldsymbol{L}$ is the *Laplacian matrix* of the graph G, which is given by

$$\boldsymbol{L} \overset{\text{def}}{=} \boldsymbol{D} - \boldsymbol{A}.$$

This last expression is

$$\frac{\gamma_2}{n},$$

where γ_2 is the second-smallest eigenvalue of the Laplacian matrix. (and will be closely related to λ_2). To help make sense out of this, let me recall a few simple facts about the Laplacian matrix (whose proofs are very similiar to things that we proved for walk matrices).

Claim 6.5.1. *If L is the Laplacian matrix of a connected graph, then*

1. *The smallest eigenvalue of L is 0.*
2. *The eigenspace of eigenvalue 0 is dimension 1.*
3. *The eigenspace of eigenvalue 0 is spanned by $\mathbf{1}$.*

These facts, and the Courant-Fischer theorem (which we will state below) may be used to establish that

$$\min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{x}\perp\mathbf{1}} \frac{\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \gamma_2.$$

6.6 The Courant-Fischer Theorem

The Courant-Fisher Theorem is the other major characterization of the eigenvalues and eigenvectors of a symmetric matrix.

Theorem 6.6.1 (Courant-Fischer). *Let A be a symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Then, for all $1 \leq k \leq n$,*

$$\lambda_k = \min_{S \text{ of dimension } k} \max_{x\in S} \frac{x^T A x}{x^T x}.$$

Note that the term

$$\frac{x^T A x}{x^T x}$$

is usually called the Rayleigh quotient of x. For an eigenvector v, the Rayleigh quotient of v is its eigenvalue.

We often exploit this theorem through the following corollary.

Corollary 6.6.2. *Let A be a symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. and corresponding eigenvectors $v_1, \ldots, v_n$. Then, for all $1 \leq k \leq n$,*

$$\lambda_k = \min_{x\perp v_1,\ldots,v_{k-1}} \frac{x^T A x}{x^T x}, \text{ and}$$

$$v_k = \arg \min_{x\perp v_1,\ldots,v_{k-1}} \frac{x^T A x}{x^T x}.$$

6.7 Relaxation of conductance

In a way that is similar to how we relaxed the minimum ratio cut problem, we may relax the minimum conductance cut problem to show that

$$\min_S r(S) \geq \min_{\boldsymbol{x} \perp \boldsymbol{d}} \frac{\boldsymbol{x}^T L \boldsymbol{x}}{\boldsymbol{x}^T D \boldsymbol{x}} = \lambda_2.$$

To make this all clear, let me define the *normalized Laplacian matrix* to be

$$\mathcal{L} \stackrel{\text{def}}{=} D^{-1/2} L D^{-1/2}.$$

We then have

$$\mathcal{L} = D^{-1/2}(D - A)D^{-1/2} = I - D^{-1/2} A D^{-1/2} = I - N,$$

where N was the normalized walk matrix. Recall that N had the same eigenvalues as the ordinary walk matrix, but it was symmetric and so more convenient to work with. This tells us that for every eigenvalue μ of the walk matrix, $1 - \mu$ is an eigenvalue of the normalized Laplacian. So, the 1 eigenvalue of the walk matrix translates into a 0 eigenvalue of the normalized Laplacian, and the reason I defined $\lambda_2 = 1 - \mu_2$, where μ_2 was the second-largest eigenvalue of the walk matrix was so that I could say that λ_2 was the second-smallest eigenvalue of the normalized Laplacian. The eigenvector of eigenvalue 0 of the normalized Laplacian is the vector that a u takes the value $\sqrt{d(u)}$. Let's call this vector $\boldsymbol{d}^{1/2}$. So, by the Courant-Fischer Theorem, we have

$$\begin{aligned}\lambda_2 &= \min_{\boldsymbol{x} \perp \boldsymbol{d}^{1/2}} \frac{\boldsymbol{x}^T D^{-1/2} L D^{-1/2} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}, \\ &= \min_{\boldsymbol{y} \perp \boldsymbol{d}} \frac{\boldsymbol{y}^T L \boldsymbol{x}}{\boldsymbol{y}^T D \boldsymbol{x}},\end{aligned}$$

by the substitution $\boldsymbol{x} = D^{1/2}\boldsymbol{y}$.

The great result relating λ_2 and conductance is called Cheeger's inequality. It was first proved for Reimanian manifolds by Cheeger in 1970. The graph-theoretic version that I state here was published by Sinclair and Jerrum in 1989. It says (roughly)

$$\Phi(G) \leq 2\sqrt{2\lambda_2}.$$

6.8 Finding a cut

Let v be the eigenvector of the second-smallest eigenvalue of $\mathcal{L}$. Recall that $w = D^{-1/2}v$ is a right-eigenvector of the second-largest eigenvalue of the walk matrix $M = D^{-1}A$.

To find a cut, we first assume that the vertices are ordered so that $v(1) \geq v(2) \geq \cdots \geq v(n)$ (We probably have to permute the vertices to achieve this). We then define the sets

$$S_j = \{1, \ldots, j\}.$$

From the proof of Cheeger's inequality, one can show that for some j,

$$\phi(S_j) \leq \sqrt{2\lambda_2(\mathcal{L})}. \tag{6.1}$$

So, while the cut we find is not necessarily optimal, it does satisfy some guarantee. In particular, (**??**) tells us that if a graph has a set of small conductance, then $\lambda_2(\mathcal{L})$ is small. Conversely, (6.1) tells us that if $\lambda_2(\mathcal{L})$ is small, then the corresponding eigenvector reveals a cut of small, but not necessarily minimum, conductance.

6.9 Disclaimer

At this point, I should mention that there are many other heuristics for finding cuts of low conductance, and there are even algorithms that are known to provide better approximations.

But, I will contintute to talk about the spectral approach because:

- it does work unusually well,
- it is easy to compute v_2 if you know a little numerical analysis, and
- I don't have all day.

6.10 Image Segmentation

Everything that I've told you so far has been known since the 80's. Around 2000, Shi and Malik rediscovered some of this material, and applied it to the segmentation of images. This introduced the technique to a new community, and generated quite a bit of interest.

Shi and Malik introduced what they called the Normalized Cut, given by

$$\min_S = \frac{\sum_{(i,j)\in E: i\in S, j\notin S} w_{i,j}}{d(S)d(V-S)},$$

and observed that a relaxation of the normalized cut was given by the normalized Laplacian.

Given an image, they considered the problem of dividing it up into regions. They did this by creating a grid graph with one vertex for each pixel in the image. They then put weights on edges of the graph whose strength was inversely related to their similarity. For example, if each pixel $p(i,j)$ is an element of $\mathbb{R}^3$, say a RGB value, then they would set the weight of the edge from (i,j) to $(i+1,j)$ to

$$e^{-\frac{(p(i,j)-p(i+1,j))^2}{\sigma^2}},$$

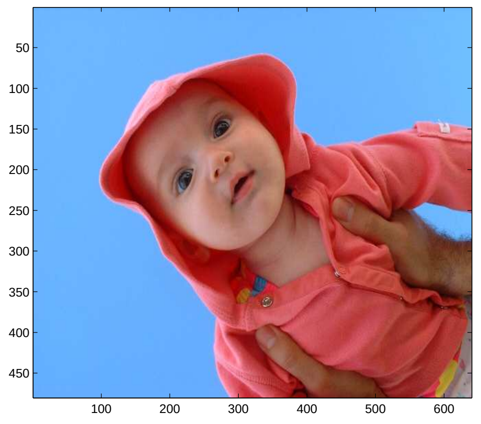
where σ is kept constant throughout the construction. So, the more different the pixels were, the lower the weight of the edge between them would be.

They then obtained eigenvectors corresponding to the top eigenvalues of the walk matrix, and used them to partition the graph. They obtained relatively good results with relatively little work.

Here is an example that I did with picture of my daugher. All the files should be listed on the class web page.

I'll cut out three-fourths of the pictures so that the size of the graph I get is more managable.

```
>> img = imread('airplane.JPG');
>> size(img)

ans =

         960        1280

>> img = img(1:2:end, 1:2:end,:)
>> size(img)

ans =

   480   640     3

```



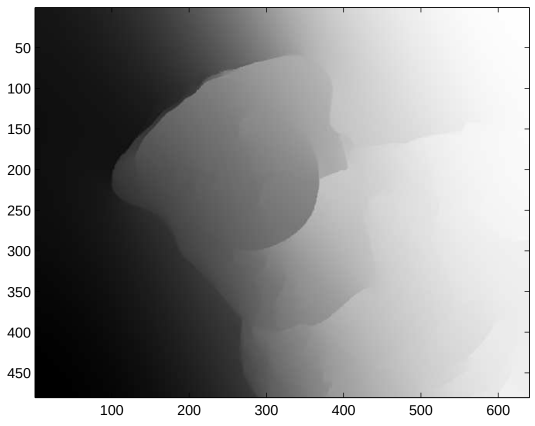
```
>> image(img)
```

The code `imgeig` returns the adjacency matrix, a, and the first k eigenvectors. The parameter *support* determines how far apart pixels can be and still be connected by an edge. As I explained that only adjacent pixels are connected, we should use support 1. In class, I actually used support 2.

```
>> [V,E,a] = imgeig(img, 6, 1);

xsz =

   480



ysz =

   640

Iteration 1: a few Ritz values of the 20-by-20 matrix:
     0
     0
     0
     0
     0
     0
     0

Iteration 2: a few Ritz values of the 20-by-20 matrix:
   1.0e+08 *

    0.0004
    0.0005
    0.0008
    0.0011
    0.0020
    0.0025
    1.0000
```

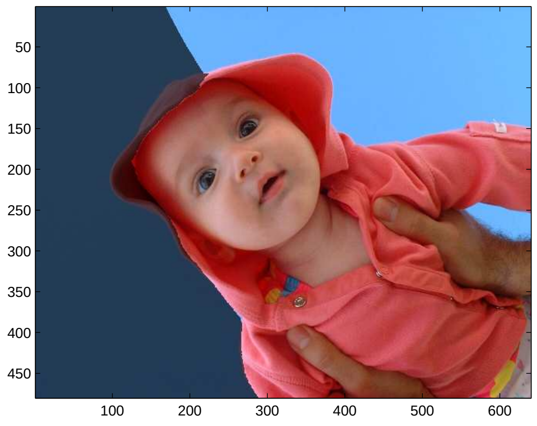
Now, let's look at the second eigenvector in greyscale.

```
>> w = V(:,2);
>> w = w-min(w);
>> w = w / max(w);
>> w = reshape(w,480,640);
>> wimg(:,:,1) = w;
>> wimg(:,:,2) = w;
>> wimg(:,:,3) = w;
>> image(wimg)
```



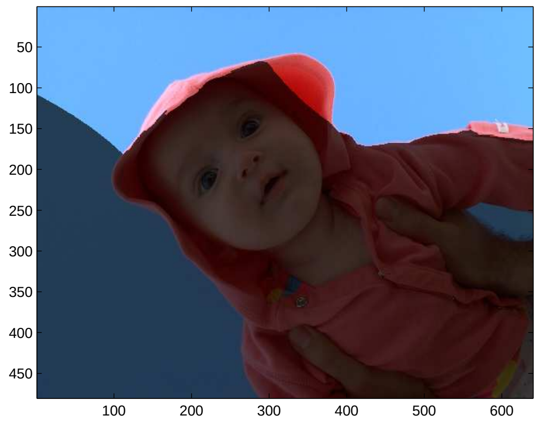
And, here is the cut generated by this eigenvector

```
>> w = w(:);
>> size(w)

ans =

      307200           1

>> S = sparsecut(a,w);
>> imcut(img,S)
```

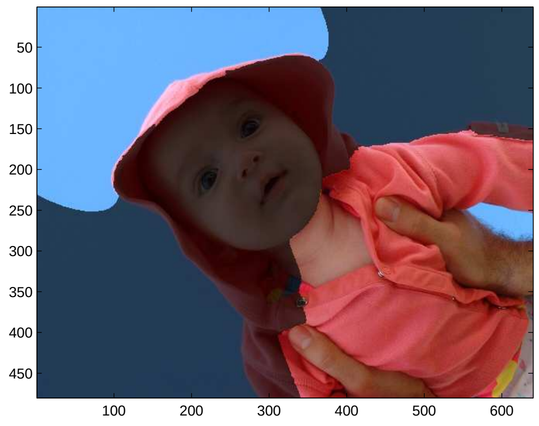


And, here are the cuts generated by the third through fifth eigenvectors.

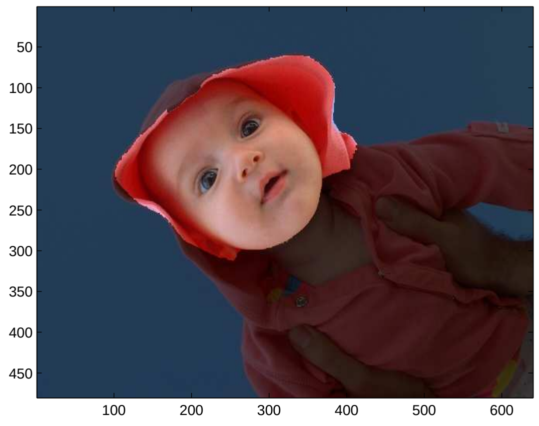
```
>> w = V(:,3);
>> S = sparsecut(a,w);
>> imcut(img,S)
```



```
>> w = V(:,4);
>> S = sparsecut(a,w);
>> imcut(img,S)
```



```
>> w = V(:,5);
>> S = sparsecut(a,w);
>> imcut(img,S)
```



I don't claim that these cuts are ideal. But, if you measured quality of cut per line of matlab, these are clear wins.