Notes on Spectral Methods for Community Detection on Graphs

Aris Anagnostopoulos

1 Some websites

https://towardsdatascience.com/spectral-clustering-aba2640c0d5b
https://juanitorduz.github.io/spectral_clustering/

2 Eigenvalues and Eigenvectors

We start with a quick reminder of eigenvectors and eigenvalues.

Definition 1. *Given a square matrix $A \in \mathbb{R}^{n\times n}$, we call the pair $(\lambda, \mathbf{v})$, with $\lambda \in \mathbb{C}$ and $\mathbf{v} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$ an (eigenvalue, (right) eigenvector) pair, if the following holds:*

$$A \cdot \mathbf{v} = \lambda \mathbf{v}.$$

To compute them, in principle we can write the definition as

$$(A - \lambda \mathbb{I})\mathbf{v} = \mathbf{0},$$

which as an equation of $\mathbf{v}$, it has n unknowns and n equations.

For this system of equations to have solutions other than $\mathbf{v} = \mathbf{0}$, the determinant of $A - \lambda\mathbb{I}$ must equal 0; this defines a polynomial in λ of degree n, which means that there are n eigenvalues (not neessarily distinct, and possibly complex numbers).

As an example, let

$$A = \begin{bmatrix} 3 & 4 \\ 1 & 6 \end{bmatrix}$$

Then we can define the matrix

$$A = \begin{bmatrix} 3-\lambda & 4 \\ 1 & 6-\lambda \end{bmatrix},$$

and we have

$$\det(A - \lambda\mathbb{I}) = (3-\lambda)(6-\lambda) - 4,$$

which is a second-degree polynomial in λ, so the matrix A has two eigenvalues.

In general the eigenvalues of a matrix can be complex number and may not all be distinct. However, we have the following important theorem from linear algebra, which we mention without proof:

Theorem 2. *If A is a symmetric matrix, then all its eigenvalues are real numbers.*

For an (eigenvalue, eigenvector) pair $(\lambda, \mathbf{v})$ of A, we have that

$$\frac{\mathbf{v}^\intercal A \mathbf{v}}{\mathbf{v}^\intercal \mathbf{v}} = \frac{\mathbf{v}^\intercal \lambda \mathbf{v}}{\mathbf{v}^\intercal \mathbf{v}} = \frac{\lambda \mathbf{v}^\intercal \mathbf{v}}{\mathbf{v}^\intercal \mathbf{v}} = \lambda. \tag{1}$$

If matrix A is symmetric, then we can choose n (not necessarily distinct) eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ eigenvalues, and n corresponding orthonormal eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$.
Recall that we call the set of n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ orthonormal if we have that

$$\mathbf{v}_i{}^\intercal \mathbf{v}_j = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Note that this allows to define them as follows. For the first one, we can write

$$\lambda_1 = \min_{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\mathbf{x}^\intercal A \mathbf{x}}{\mathbf{x}^\intercal \mathbf{x}} \qquad \mathbf{v}_1 = \operatorname*{argmin}_{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\mathbf{x}^\intercal A \mathbf{x}}{\mathbf{x}^\intercal \mathbf{x}} \tag{2}$$

for the second one,

$$\lambda_2 = \min_{\substack{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ \mathbf{x} \perp \mathbf{v}_1}} \frac{\mathbf{x}^\intercal A \mathbf{x}}{\mathbf{x}^\intercal \mathbf{x}} \qquad \mathbf{v}_2 = \operatorname*{argmin}_{\substack{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ \mathbf{x} \perp \mathbf{v}_1}} \frac{\mathbf{x}^\intercal A \mathbf{x}}{\mathbf{x}^\intercal \mathbf{x}}$$

and for general k

$$\lambda_k = \min_{\substack{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ \mathbf{x} \perp \mathbf{v}_1, \ldots, \mathbf{x} \perp \mathbf{v}_{k-1}}} \frac{\mathbf{x}^\intercal A \mathbf{x}}{\mathbf{x}^\intercal \mathbf{x}} \qquad \mathbf{v}_k = \operatorname*{argmin}_{\substack{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ \mathbf{x} \perp \mathbf{v}_1, \ldots, \mathbf{x} \perp \mathbf{v}_{k-1}}} \frac{\mathbf{x}^\intercal A \mathbf{x}}{\mathbf{x}^\intercal \mathbf{x}}.$$

Incidentally, assume that $\lambda_1 = \lambda_2$, meaning that there are two orthogonal eigenvectors, $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to the same eigenvalue. Then note that every linear combination of $\mathbf{v}_1$ and $\mathbf{v}_2$ is also an eigenvector corresponding to $\lambda_1 = \lambda_2$:

$$A(c_1\mathbf{v}_1 + c_2\mathbf{v}_2) = c_1 A\mathbf{v}_1 + c_2 A\mathbf{v}_2 = c_1\lambda_1\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 = c_1\lambda_1\mathbf{v}_1 + c_2\lambda_1\mathbf{v}_2 = \lambda_1(c_1\mathbf{v}_1 + c_2\mathbf{v}_2).$$

But typically, we will choose $\mathbf{v}_1$ and $\mathbf{v}_2$ to be unit vectors, orthogonal to each other.

3 The Adjacency and the Laplacian Matrices

We are given a simple undirected graph $G = (V, E)$. Many of the things of these notes hold also for weighted graphs, but for simplicity we assume an unweighted one. Recall that its *adjacency matrix* $A \in \{0, 1\}^{n \times n}$ is the matrix that indicates the edges:

$$A_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise.} \end{cases}.$$

The adjacency matrix of a graph is very important as it captures the entire graph information. There exists a transformation of the adjacency matrix, the Laplacian matrix, which for some problems is easier to work with. Let's see it now.

First, we define $[n] = \{1, 2, \ldots, n\}$, and we let the set of nodes be $V = [n]$. For each node $i \in [n]$, let d_i be the degree of node i.

Then we can define the diagonal matrix D, as:

$$D_{ij} = \begin{cases} d_i, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}.$$

The matrix D is invertible if and only if $d_i > 0$ for every $i \in [n]$. Then we have

$$D^{-1}_{ij} = \begin{cases} \frac{1}{d_i}, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}.$$

We define D^{-1} even if there are nodes with degree 0, with the convention that $D^{-1}_{ii} = 0$ if $d_i = 0$.

We then define $D^{-1/2}$ to be the matrix

$$D^{-1/2}_{ij} = \begin{cases} \frac{1}{\sqrt{d_i}}, & \text{if } i = j \text{ and } d_i > 0 \\ 0, & \text{otherwise.} \end{cases}.$$

We define the *Laplacian matrix* of graph G to be the matrix

$$L = D - A.$$

Then note that we have that

$$L_{ij} = \begin{cases} d_i, & \text{if } i = j \\ -1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}.$$

We will see that the Laplacian matrix provides us a lot of information about the connectivity of G, and it can be used to discover communities!

It is often useful to define the *normalized Laplacian matrix* $\mathcal{L}$. This is defined as the matrix:

$$\mathcal{L}_{ij} = \begin{cases} 1, & \text{if } i = j \\ -\frac{1}{\sqrt{d_i d_j}}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}.$$

Then one can check that it can be written as

$$\mathcal{L} = \mathbb{I} - D^{-1/2} A D^{-1/2}.$$

Often in these notes we will work with d-regular graphs for simplicity. A graph is *d-regular* if every node has degree d. Then we have that

$$D = \frac{1}{d}\mathbb{I},$$

and the normalized Laplacian can be written as

$$\mathcal{L} = \mathbb{I} - \frac{1}{d}A.$$

We will now prove a very useful property of the Laplacian matrix.

Theorem 3. *Consider a graph $G = (V, E)$ and let L be its Laplacian matrix. Let $\mathbf{x} = [x_1, \ldots, x_n]^\intercal$ be any real-valued matrix. Then we have that*

$$\mathbf{x}^\intercal L\mathbf{x} = \sum_{(i,j)\in E} (x_i - x_j)^2.$$

Proof. We have

$$\mathbf{x}^\intercal L\mathbf{x} = \mathbf{x}^\intercal (D - A)\mathbf{x} = \mathbf{x}^\intercal D\mathbf{x} - \mathbf{x}^\intercal A\mathbf{x}.$$

We will look at each of these two terms separately.

$$\begin{aligned}\mathbf{x}^\intercal D\mathbf{x} &= \begin{bmatrix} x_1, \ldots, x_i, \ldots, x_n \end{bmatrix} \cdot \begin{bmatrix} d_1 & & & & \\ & \ddots & & \mathbf{0} & \\ & & d_i & & \\ & \mathbf{0} & & \ddots & \\ & & & & d_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \\ &= \begin{bmatrix} d_1x_1, \ldots, d_ix_i, \ldots, d_nx_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} d_i x_i^2 \end{aligned} \tag{3}$$

Let $\mathbf{A}_{(i)}$ be the ith row of A. Also define $\mathbf{x}_i = [0, \ldots, 0, x_i, 0, \ldots, x_n]^\intercal$, such that $\mathbf{x} = \sum_{i=1}^{n} \mathbf{x}_i$. We then have:

$$\begin{aligned}{\mathbf{x}_i}^\intercal A\mathbf{x} &= \begin{bmatrix} 0, \ldots, x_i, \ldots, 0 \end{bmatrix} \cdot \begin{bmatrix} \text{———}\mathbf{A}_{(1)}\text{———} \\ \vdots \\ \text{———}\mathbf{A}_{(i)}\text{———} \\ \vdots \\ \text{———}\mathbf{A}_{(n)}\text{———} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \\ &= \begin{bmatrix} \text{———}x_i\mathbf{A}_{(i)}\text{———} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \sum_{j\in N(i)} x_i x_j, \end{aligned}$$

where $N(i)$ is the neighborhood of i. This is because the jth element of vector $x_i\mathbf{A}_{(i)}$ equals x_i if $j \in N(i)$ and 0 otherwise.

$$\mathbf{x}^\intercal A\mathbf{x} = \left(\sum_{i=1}^{n} {\mathbf{x}_i}^\intercal\right) A\mathbf{x} = \sum_{i=1}^{n} ({\mathbf{x}_i}^\intercal A\mathbf{x}) = \sum_{i=1}^{n} \sum_{j\in N(i)} x_i x_j = 2 \sum_{(i,j)\in E} x_i x_j, \tag{4}$$

because in the second-to-last summation, each pair (i, j) is counted twice if $(i, j) \in E$ and zero times if $(i, j) \notin E$.

Combining Equations (3) and (4), we obtain

$$\mathbf{x}^{\intercal} L \mathbf{x} = \sum_{i=1}^{n} d_i x_i^2 - 2 \sum_{(i,j)\in E} x_i x_j. \tag{5}$$

On the other hand, we have

$$\sum_{(i,j)\in E} (x_i - x_j)^2 = \sum_{(i,j)\in E} \left(x_i^2 + x_j^2 - 2x_i x_j\right) = \sum_{(i,j)\in E} \left(x_i^2 + x_j^2\right) - 2 \sum_{(i,j)\in E} x_i x_j.$$

Note that in the left summation, each edge is counted once. This means that for each edge $(i, j) \in E$ each endpoint appears once (either as i or as j). Therefore, the number of times that a node is represented in the summation (either as x_i or as x_j) is equal to the set of adjacent edges, that is, its degree. Therefore we obtain

$$\sum_{(i,j)\in E} (x_i - x_j)^2 = \sum_{i=1}^{n} d_i x_i^2 - 2 \sum_{(i,j)\in E} x_i x_j. \tag{6}$$

Combining Equations (5) and (6), completes the proof of the theorem. □

4 Graph Cuts

A graph cut (C_1, C_2) is simply a partitioning of the nodes in two sets C_1, C_2. That is: $C_1 \cap C_2 = \emptyset$ and $C_1 \cup C_2 = V$.

We define the size of the cut (C_1, C_2) to be the number of edges between the two sets C_1 and C_2. Let $E(C_1, C_2)$ be the set of edges between C_1 and C_2:

$$E(C_1, C_2) = E \cap (C_1 \times C_2) = \{(i, j) \in E : i \in C_1, j \in C_2\}.$$

Then the size of the cut (C_1, C_2) is $|E(C_1, C_2)|$.

Intuitively, if we want to split a graph into communities, we want to partition it where the cut size is small. Thus, it is very natural to define the problem of finding a minimum cut:

Definition 4 (MinCut)**.** *The* mincut *problem is the problem of finding a cut (C_1, C_2) that minimizes $|E(C_1, C_2)|$.*

Finding a minimum cut can be done in polynomial time. There exist algorithms that are based on the concept of *flow*. There also exists a very elegant randomized algorithm (Karger's algorithm).

Unfortuntately, this definition is not very useful for finding communities (why?). Therefore, we next consider the problem of sparset cut.

4.1 The Sparsest-Cut Problem

The problem with MinCut is that it does not take into account the number of notdes in each partition. Very often, C_1 may be just one node! Therefore, there exist various other ways to find cuts, which take into account the number of the nodes in C_1 and C_2, and try, not only to induce cuts of small size, but also cuts that partition the edges into large groups. Here we will consider one of them, the sparsest cut.

Definition 5 (SparsestCut)**.** *The* sparset cut *is a cut* (C_1, C_2) *that minimizes*

$$\min_{\substack{C_1 \subset V \\ C_2 = V \setminus C_1}} \frac{|E(C_1, C_2)|}{\min\{|C_1|, |C_2|\}}.$$

Note that the denominator in the objective function favors cuts in which both C_1 and C_2 are as large as possible, so the sparsest-cut problem tries to partition the graph into two, in a way that the edges between the two parts are as small as possible, and at the same time each of the two parts is as large as possible.

Unfortunately, the problem of computing the sparsest cut is NP-hard. Furthermore, no constant approximation is known. However, we will see how it is related to the graph structure, and how we can come up with an approximation algorithm in which the approximation ratio depends on the graph structure.

To do this we will provide an algebraic formulation of the problem, that is, we will express it in terms of matrices. Let $\mathbf{x} \in \{0,1\}^n$ be an indicator vector for set C_1, that is,

$$x_i = \begin{cases} 1, & \text{if } i \in C_1 \\ 0, & \text{otherwise} \end{cases}.$$

Also let $C_2 = V \setminus C_1$. Then first note that we have

$$\mathbf{x}^\mathsf{T}\mathbf{x} = \sum_{i=1}^{n} x_i^2 = |C_1| \tag{7}$$

Also, by Equation (3) we have that

$$\mathbf{x}^\mathsf{T} D\mathbf{x} = \sum_{i \in C_1} d_i = 2\,|E \cap (C_1, C_1)| + |E \cap (C_1 \times C_2)|\,, \tag{8}$$

and by Equation (4) we have that

$$\mathbf{x}^\mathsf{T} A\mathbf{x} = 2\,|E \cap (C_1 \times C_1)| \tag{9}$$

Combining Equations (8) and (9), we obtain that

$$\mathbf{x}^\mathsf{T} L\mathbf{x} = |E \cap (C_1 \times C_2)| = |E(C_1, V \setminus C_1)|\,, \tag{10}$$

that is, the size of the cut (C_1, C_2).

Now, note that the smallest of the two sets C_1 and C_2 has size at most $n/2$ and the largest at least $n/2$, so, using Equations (7) and (10), we can write

$$\min_{\substack{C_1 \subset V \\ C_2 = V \setminus C_1}} \frac{|E(C_1, C_2)|}{\min\{|C_1|, |C_2|\}} = \min_{\substack{C_1 \subset V \setminus \emptyset \\ |C_1| \le \frac{n}{2}}} \frac{|E(C_1, V \setminus C_1)|}{|C_1|} = \min_{\substack{\mathbf{x} \in \{0,1\}^n \setminus \{\mathbf{0}\} \\ \|\mathbf{x}\| \le \frac{n}{2}}} \frac{\mathbf{x}^\mathsf{T} L\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{x}}.$$

Note the similarity between this formulation of the sparsest cut, and the expression in Equation (2). It turns out that this is very useful and we will take advantage of it.

4.2 Graph Conductance

A concept related to sparsity is the *conductance.*

Definition 6. *Given a graph $G = (V, E)$, and a partition $(C_1, V \setminus C_1)$, we define the conductance $\phi(C_1)$ to be*

$$\phi(C_1) = \frac{|E(C_1, C_2)|}{\min\left\{\sum_{i \in C_1} d_i, \sum_{i \in C_2} d_i\right\}}.$$

The conductance of a graph ϕ_G is defined as the minimum conductance among all its partitions.

Definition 7. *Given a graph $G = (V, E)$, its conductance ϕ_G is*

$$\phi_G = \min_{C_1 \subset V} \phi(C_1) = \min_{\substack{C_1 \subset V \\ C_2 = V \setminus C_1}} \frac{|E(C_1, C_2)|}{\min\left\{\sum_{i \in C_1} d_i, \sum_{i \in C_2} d_i\right\}}.$$

The conductance of a graph tells us a lot about its connectivity. For example, it quantifies how much time it will require for a random walk on the graph to converge to a stationary distribution.

Note that if G is d-regular, we have that

$$\phi_G = \min_{\substack{C_1 \subset V \\ C_2 = V \setminus C_1}} \frac{|E(C_1, C_2)|}{d \cdot \min\{|C_1|, |C_2|\}}.$$

Therefore, in regular graphs, minimizing the conductance, is equivalent to minimizing the sparsest cut!

4.3 Eigenvalues of the Laplacian

We now start by showing some of the properties of G that its eigenvalues reveal.

Theorem 8. *Let G be an undirected d-regular graph, with normalized Laplacian $\mathcal{L}$. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of $\mathcal{L}$. Then the following are true:*

1. $\lambda_1 = 0$

2. $\lambda_k = 0$ *if and only if G has at least k connected components.*

Proof. We first prove part 1. By Equation (2) we have that

$$\lambda_1 = \min_{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\mathbf{x}^\mathsf{T} \mathcal{L} \mathbf{x}}{\mathbf{x}^\mathsf{T} \mathbf{x}}.$$

By Theorem 3, we obtain that

$$\lambda_1 = \min_{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\mathbf{x}^\mathsf{T} \mathbf{x}}.$$

This equation, immediately gives that $\lambda_1 \geq 0$. Next, notice that the ratio

$$\frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\mathbf{x}^\mathsf{T} \mathbf{x}}.$$

is equal to 0 for $\mathbf{x} = \mathbf{1}$. This proves that $\lambda_1 = 0$.

For the second part, assume that G has exactly k components. Then we will prove that $\lambda_k = 0$ and that $\lambda_{k+1} > 0$, which proves the claim.

Let $C_1, \dots, C_k$ be the k components of G, that is, $V = C_1 \cup \cdots \cup C_k$. For $\ell \in [k]$, we define the k vectors $\mathbf{x}^\ell$, defined as

$$x_i^\ell = \begin{cases} 1, & \text{if } i \in C_\ell \\ 0, & \text{otherwise} \end{cases}.$$

In other words, $\mathbf{x}^\ell$ is the indicator vector of component C_ℓ.

Note two things. First, the k vectors $\mathbf{x}^\ell$ are orthogonal to each other. This is because for each ℓ, r with $\ell \neq r$ we have $C_\ell \cap C_r = \emptyset$, so $(\mathbf{x}^\ell)^\mathsf{T} \mathbf{x}^r = 0$. Second, for each $\mathbf{x}_\ell$, we have that

$$\frac{\sum_{(i,j)\in E} (x_i^\ell - x_j^\ell)^2}{(\mathbf{x}^\ell)^\mathsf{T} \mathbf{x}^\ell} = \frac{\sum_{r=1}^k \sum_{(i,j)\in E\cap(C_r\times C_r)} (x_i^\ell - x_j^\ell)^2}{(\mathbf{x}^\ell)^\mathsf{T} \mathbf{x}^\ell} = 0.$$

The first equality, follows from the fact that $C_1, \dots, C_k$ are the connected component of G, and it just expresses the fact that all the edges E are the edges inside each of the conneted components. The second inequality follows from the definition of each $\mathbf{x}^\ell$: for two nodes i, j inside a particular component C_r we have $x_i^\ell = x_j^\ell (= 0 \text{ or } 1$, based on whether $r = \ell)$. Therefore we have k mutually orthogonal vectors $\mathbf{x}^\ell$ for which the ratio

$$\frac{(\mathbf{x})^{\ell\mathsf{T}} \mathcal{L} \mathbf{x}^\ell}{(\mathbf{x})^{\ell\mathsf{T}} \mathbf{x}^\ell} = \frac{\sum_{(i,j)\in E} (x_i^\ell - x_j^\ell)^2}{(\mathbf{x}^\ell)^\mathsf{T} \mathbf{x}^\ell} = 0,$$

and, therefore, we have that $\lambda_1 = \cdots = \lambda_k = 0$.

Note: In the proof of the first part of this theorem, we set $\mathbf{v}_1 = \mathbf{1}$. Here, instead, we set $\mathbf{v}_1$ to be the indicator vector of component C_1. Note that we can write $\mathbf{1} = \sum_{\ell=1}^k \mathbf{x}^\ell$. Therefore, the vector $\mathbf{1}$ *is in the span of the vectors* $\mathbf{v}_1, \dots, \mathbf{v}_\ell$. The eigenvectors are not unique, so we can choose the ones that are more convenient for us. For the proof of the first part it was convenient to choose $\mathbf{v}_1 = \mathbf{1}$, instead for the second part it was convenient to take another choice. But one part does not contradict the other.

To show that $\lambda_{k+1} > 0$ we will show that any vector $\mathbf{x}$ for which

$$\frac{\mathbf{x}^\mathsf{T} \mathcal{L} \mathbf{x}}{\mathbf{x}^\mathsf{T} \mathbf{x}} = \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\mathbf{x}^\mathsf{T} \mathbf{x}} = 0$$

can be written as a linear combination of the k vectors $\mathbf{x}^\ell$. We have written the numerator as

$$\sum_{r=1}^k \sum_{(i,j)\in E\cap(C_r\times C_r)} (x_i^\ell - x_j^\ell)^2.$$

Because each of the terms are nonnegative, for the summation to be zero, it is required to have $x_i = x_j$ for each i, j in the same connected component. To see this, assume that this is not the case, that is, there exists a connected component C_m and two nodes $s, t \in C_m$ with $x_s \neq x_t$. Because s and t are in the same connected component, there exists a path P between them. Then we have that

$$\sum_{r=1}^k \sum_{(i,j)\in E\cap(C_r\times C_r)} (x_i^\ell - x_j^\ell)^2 \geq \sum_{(i,j)\in E\cap(C_m\times C_m)} (x_i^\ell - x_j^\ell)^2 \geq \sum_{(i,j)\in P} (x_i^\ell - x_j^\ell)^2.$$

The first inequality is true because C_m is one of the k components C_r, and the second one is because all the edges in P belong to C_m. But s and t are the two endpoints of the path P and, because we have assumed that $x_s \neq x_t$, it means that somewhere in the path there exists an edge $(i,j) \in P$ with $x_i \neq x_j$. For this edge we have that $(x_i - x_j)^2 > 0$, which would imply that the entire ratio

$$\frac{\sum_{(i,j)\in E}(x_i - x_j)^2}{\mathbf{x}^\intercal \mathbf{x}} > 0.$$

This is a contradiction. Therefore, we have that for each component C_r, and for each $i \in C_j$ the value x_i is constant. This means that $\mathbf{x}$ can be written as a linear combination of the k vectors $\mathbf{x}^\ell$.

Another way to say this, is that if $\mathbf{x}$ is orthogonal to all the k vectors $\mathbf{x}^\ell$ then

$$\frac{\sum_{(i,j)\in E}(x_i - x_j)^2}{\mathbf{x}^\intercal \mathbf{x}} > 0,$$

which implies that we have $0 < \lambda_{k+1} \leq \cdots \leq \lambda_n$.

This complete the proof of the theorem. □

Let us see what this theorem tells us. It says, in particular, that $\lambda_2 = 0$ if and only if the graph is disconnected. Notice, also, that if the graph is disconnected, then we also have $\phi_G = 0$. So, in this case, we have that $\phi_G = \lambda_2$.

Note that computing the eigenvalues of a matrix is an easy problem (not NP-hard). On the other hand, we said that computing the conductance ϕ_G is NP-hard. It would then be nice to be able to relate more generally ϕ_G with λ_2. Indeed the two quantities are related, and one is an approximation of the other. This is formalized by *Cheeger's inequality*:

Theorem 9 (Cheeger's inequality)**.** *Let $G = (V,E)$ be an undirected graph, with conductance ϕ_G, and with normalized Laplacian matrix $\mathcal{L}$, whose second smallest eigenvalue is λ_2. Then we have that:*

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

The proof of this theorem has two parts, one for each inequality. We will not prove it here. However we note that the proof of the second inequality is constructive, that is, it provides an algorithm that allows to find a cut that approximates the conductance ϕ_G.

```
1. Function SWEEPING($G$)
2. Input: $G = (V,E)$: Simple undirected $d$-regular graph with normalized Laplacian matrix $\mathcal{L}$
3. Output: A cut $(C_1, C_2)$ of $G$
4.    Compute the second smallest eigenvalue $\lambda_2$ of $\mathcal{L}$ and the corresponding eigenvector
      $\mathbf{x} = \mathbf{v}_2$.
5.    Sort the vertices of $V$ in increasing order of $x_i$. Assume that $x_1 \leq x_2 \leq \cdots \leq x_n$.
6.    For each $i \in [n-1]$ consider the cut $(C_1, C_2)$ with $C_1 = [i]$ and $C_2 = V \setminus C_1$. Compute
      the value
          $\phi(C_1) = \frac{|E(C_1, C_2)|}{d \cdot \min\{|C_1|, |C_2|\}}$.
7.    From all the $n-1$ cuts computed in step 6, return the one that minimizes $\phi(C_1)$.
```

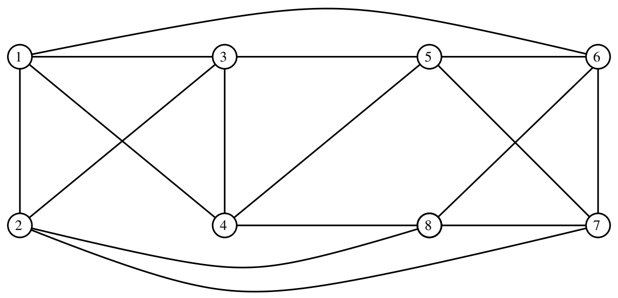

Figure 1: Example for sparsest cut and Cheeger's inequality

The proof of Theorem 8 can give us some intuition about why the second eigenvalue allows us to partition the graph into two. Assume, for simplicity, that G is formed by 5 nodes, and two connected components: $\{1,2,3\}$ and $\{4,5\}$. By Theorem 8, $\lambda_1 = 0$, and assume that we obtain $\mathbf{v}_1 = [1,1,1,1,1]^\intercal$. Because the graph is disconnected, we also have that $\lambda_2 = 0$. Furthermore, by the proof of Theorem 8, we see that in $\mathbf{v}_2$, the values corresponding to each node of the same component are the same. In other words, we will have that $\mathbf{v}_2 = [\alpha,\alpha,\alpha,\beta,\beta]^\intercal$ for two distinct constants α, β. Thus, $\mathbf{v}_2$ reveals the structure of the graph and the algorithm will discover the two components.

This is an extreme case. More generally, if the graph is connected, if there are two dense clusters connected by a small number of edges, then $\mathbf{v}_2$ will not separate the two clusters perfectly, but it will still be able to assign to the coordinates of $\mathbf{v}_2$ nearby values for nodes that belong to the same cluster and far for nodes that belong to different clusters. Furthermore, from Cheeger's inequality we obtrain that λ_2 is positive but small. Thus, looking at the components of $\mathbf{v}_2$, we can identify the two communities.

Assume that we want to partition the graph into more than two communities. Then we can partition the graph into two, and thun recur. Another approach is what we discuss in the next section.

5 Example

In Figure 1 you can see a 4-regular graph. Verify Cheeger's inequality and find the sparsest cut given by algorithm Sweeping.

```
A = [0, 1, 1, 1, 0, 1, 0, 0; 1, 0, 1, 0, 0, 0, 1, 1; 1, 1, 0, 1, 1, 0, 0, 0; 1,
0, 1, 0, 1, 0, 0, 1; 0, 0, 1, 1, 0, 1,1, 0; 1, 0, 0, 0, 1, 0, 1, 1; 0, 1, 0, 0, 1,
1, 0, 1; 0, 1, 0, 1, 0, 1, 1, 0]
sum(A)
sum(A')
A-A'
D = eye(8)
L = eye(8) - A/4
sum(L)
eig(L)
[X, Y] = eig(L)
```

```
X(:,1)
r=X(:,1)
r' * r
r=X(:,2)
sum(r)
6/16
sqrt(6.4645e-01 * 2)
6.4645e-01 /2
v2=X(:,2)
[out,idx] = sort(v2)
```

6 Spectral Clustering

Until now we have seen that:

- If the graph has two connected components, then we have that $\lambda_1 = \lambda_2 = 0$ and that $\mathbf{v}_2$ can be used to discover the two components.
- We can generalize this intuition, and we can use $\mathbf{v}_2$ to discover two dense communities, even if the graph is not disconnected. The smaller the value of λ_2, the more distinct the two communities will be.

But Theorem 8 takes this one step further:

- If the graph has k connected components, then we have that $\lambda_1 = \cdots = \lambda_k = 0$ and that $\mathbf{v}_1, \ldots \mathbf{v}_k$ can be used to discover the k components.

Thus, one may ask whether in the common case where the graph is not disconnected, vectors $\mathbf{v}_3, \ldots, \mathbf{v}_k$ can provide additional information for finding k communities.

The answer is yes, and there have been various different approaches that use this idea. They are similar but they differ in some details. Here we present one of this variants, which we call SpectralClustering.

```
1. Function SpectralClustering$(G, k)$
2. Input: $G = (V, E)$: Simple undirected $d$-regular graph with normalized Laplacian matrix $\mathcal{L}$
   and a positive integer $k$
3. Output: A $k$-partition $(C_1, C_2, \ldots, C_k)$ of $G$
4.    Compute the $k$ smallest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$ of $\mathcal{L}$ and the corresponding eigenvectors
      $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.
5.    Define the $n \times k$ matrix $U$ whose columns are the vectors $\mathbf{v}_i$:
          $U = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_k \end{bmatrix}$
6.    For each $i \in [n]$, define the row vector $\mathbf{y}_i \in \mathbb{R}^{1 \times n}$ with $\mathbf{y}_i = [(v_1)_i, (v_2)_i, \ldots, (v_k)_i]$. In
      other words, associate each node $i \in V$ with the $i$th row of U
7.    Run $k$-means on the $n$ points $\{\mathbf{y}_i\}_{i=1}^n$, and obtain $k$ clusters $Y_1, Y_2, \ldots, Y_k$
8.    for $\ell = 1$ to $k$:
9.        $C_\ell = \{i \in [n] : \mathbf{y}_i \in Y_\ell\}$
10.   return $(C_1, \ldots, C_k)$
```

We can think of the vector $\mathbf{y}_i$ as the *embedding* of node i on $\mathbb{R}^k$. We will discuss about node and graph embeddings more in the future.

There are various variants of this algorithm. For example, because $\mathbf{v}_1$ is the constant vector, we can omit it and instead consider the eigenvectors $\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_{k+1}$. Another suggestion is to normalize the points and thus, after step 6, replace each point $\mathbf{y}_i$ with $\frac{\mathbf{y}_i}{\|\mathbf{y}_i\|}$.