Network analysis preliminaries: Graph theory and linear algebra

Kyle Kloster

January 9, 2017

Abstract

Basic graph and matrix definitions, ideas, tools and facts.

1 Overview

This document summarizes the relevant terminology, ideas, and tools to dig into some network analysis. We're particularly interested in graph structures and sparsity (like cliques and k-cores), and their relationship to graph-matrix objects (like functions of the adjacency matrix and eigenvalues of the Laplacian). We'll define and discuss these terms (and many more) in the sections to come.

As with many subjects there is a wealth of terminology to cover at the start – if it feels overwhelming, or unmotivated, don't worry. It is abstract and there is a bit much of it, but as we use the definitions and notation more and more, it will become easier. All of the terms here will be re-used a lot as we study and discuss objects and problems with neat theory and applications.

2 Notation and definitions

2.1 Graph basics

Throughout, let $G = (V, E)$ be a graph with nodes (vertices) V and edges (links) E. We let $n = |V|$ and $m = |E|$. We label the nodes with integers 1 through n and can refer to a node v_j by its integer label j. We identify an edge by its endpoints i, j, and label it e_{ij}. The *degree* of a node is the number of edges touching that node, and we denote it $d(j)$ or d_j. A graph is *simple* if no nodes have more than one edge between them (i.e. no multi-edges). A graph is *loopless* if no node has a *self edge* (an edge that has the same node for both of its endpoints). A graph is *undirected* if no edge has a direction, and is *directed* otherwise. A graph is *weighted* if either the nodes or edges have scalars $e \in \mathbb{R}$ associated with them; a graph is *unweighted* otherwise (in which case all edges are assumed to have weight 1).

Unless otherwise stated, we deal entirely with graphs that are simple, loopless, undirected, and unweighted. Because this setting is the one most often considered by most studies, we will call such a graph a *standard* graph. We will also usually deal with graphs that are *connected* (defined below).

2.2 Connectivity basics

A *walk* from node i to node j is a sequence of nodes starting with s and ending with t such that consecutive nodes in the sequence, v_j and v_{j+1}, are connected by an edge. Intuitively: imagine standing on a node y in G, then picking an edge that touches your node, and stepping across that edge. Then repeat. Any node v that you land on, you have arrived via a walk from y to v.

Note that a walk from s to t is allowed to step across any edge or node multiple times. A *path* is a walk that does not repeat any nodes, except possibly the final node, i.e. a path could begin at node v and ends at node v. Any walk or path that begins and ends at the same node is called *closed*.

A graph G is *connected* if for every pair of nodes u, v in V there exists a path between nodes u and v. Any graph G can be divided into *connected components* $G_1, \cdots, G_c$, i.e. subgraphs of G such that all nodes of G are contained in the node-sets of G_j, and each subgraph G_j is a connected graph. (Then no edges would connect nodes in subgraphs G_i, G_j with $i \neq j$.) The graph consisting of a single node is considered a single connected component.

An *edge cut-set* S of a graph G is a set of edges such that if remove all edges from G, the resulting graph G' will one more connected component than G. In particular, if G is a connected graph, then an edge cut-set S is a set of edges such that deleting those edges will separate (split, disconnect) G into two connected components.

2.3 Matrix and vector basics

In general we use non-bold letters $(a, b, c, \lambda, x, \gamma)$ to represent scalars, and bold letters to denote arrays/vectors/matrices. Lower-case bold letters indicate vectors, e.g. $\mathbf{e}, \mathbf{v}, \mathbf{x}, \boldsymbol{\lambda}$, whereas upper-case bold letters indicate matrices, $\mathbf{A}, \mathbf{M}, \mathbf{X}, \boldsymbol{\Lambda}$. To refer to a specific vector entry, we might use any of $\mathbf{x}(j), \mathbf{x}[j], (\mathbf{x})_j, x(j), x[j], x_j$. To refer to entry i, j of a matrix $\mathbf{A}$, we use $A(i, j)$, $A_{i,j}$, or $(\mathbf{A})_{i,j}$. We denote the transpose of a matrix $\mathbf{M}$ by $\mathbf{M}^T$.

The dimension of any vector and matrix can usually be inferred from the context if it is not specifically stated. When dealing with a graph on n nodes, most vectors are $n \times 1$ (i.e. we use column vectors) and most matrices are $n \times n$, though occasionally there are matrices whose dimensions are determined by some quantity other than the number of nodes, for instance the number of edges. There are a handful of specific letters that we reserve for specific objects: the vector $\mathbf{e}$ denotes the vector of all 1s; the vectors $\mathbf{e}_j$ are standard basis vectors, meaning they are all 0s except with a 1 in the jth entry. The matrix $\mathbf{I}$ is the $n \times n$ identity matrix, 0 is used to denote the zero vector and the zero matrix, and $\mathbf{J}$ denotes a matrix of all 1s.

To any n-node graph $G = (V, E)$ we can associate the graph's *adjacency matrix*, usually denoted $\mathbf{A}$: it is an $n \times n$ matrix in which row j and column j each encode the edge-information of node j. More specifically, the matrix satisfies $\mathbf{A}_{i,j} = 1$ iff nodes i and j are connected by an edge. Then $\mathbf{A}_{i,j} = 0$ otherwise. Note that this matrix is symmetric, since entries $\mathbf{A}_{i,j}$ and $\mathbf{A}_{j,i}$ are both "1" when nodes i and j are connected by an edge.

We use $\mathbf{d}$ to denote the $n \times 1$ vector of degrees, so that $\mathbf{d}_j = d(j)$, and we use $\mathbf{D}$ to denote the diagonal *degree matrix*, $\mathbf{D} = \text{diag}(\mathbf{d})$. Then we can define the *Laplacian matrix* (also called the *combinatorial Laplacian*, *discrete Laplacian*, and *Kirchoff matrix*) to be $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

The *edge-node incidence matrix* (which we call simply the *incidence matrix*) of a graph that has m edges and n nodes is the $m \times n$ matrix $\mathbf{B}$ that has $\mathbf{B}(i, j) = \pm 1$ iff edge i has node j as one of its endpoints. To be more precise about the sign of the entries, let edge i have node endpoints j_1 and j_2 and label those nodes so that $j_1 < j_2$. Then $\mathbf{B}(i, j_1) = -1$ and $\mathbf{B}(i, j_2) = 1$. You can think of this as every edge pointing from its endpoint with smaller label to its endpoint with larger label, so that the edge is negative (outgoing) at the smaller label j_1 and positive (ingoing) at the larger label node.

Don't get caught up in these details, but to be totally correct, these details ought to be mentioned: (Note that we know the labels of the two nodes are not equal, because we assumed our graph is loopless, and so no node can be both endpoints of a single edge. Since the edge endpoints are distinct nodes, we know one of the two must have a larger label, and hence we can assume $j_1 < j_2$). Also, this particular way of assigning signs to the entries of matrix $\mathbf{B}$ is one coherent way to assign ± 1 entries, but it turns out you can assign the signs in almost any way, as long as each edge has exactly one positive and one negative endpoint, and the important results will all still hold for the matrix $\mathbf{B}$.

2.4 Exercises

2.1 Prove that for any standard graph, the sum of the degrees of all nodes equals twice the number of edges, i.e. $\sum_{v \in V} d(v) = 2|E|$.

2.2 Let G be a standard graph with $n > 1$ nodes. Is it possible that all n nodes have distinct degrees, i.e. is it possible that no two nodes have the exact same degree? If yes, give an example, if no prove that it is impossible.

2.3 Let G be a connected, standard graph. Prove that deleting a single edge, e, from G cannot disconnected G into more than two connected components. In other words, suppose that deleting e from G produces the disconnected graph $G' = G - \{e\}$; show the number of connected components is 2.

2.4 Let G be a standard graph with adjacency matrix $\mathbf{A}$. Prove that $\mathbf{d} = \mathbf{A}\mathbf{e}$. Prove that $\mathbf{e}^T\mathbf{A}\mathbf{e} = 2|E|$.

2.5 Let G be a standard graph with Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Prove that $\mathbf{L}\mathbf{e} = 0$ using problem 2.4.

2.6 Let G be a standard graph with incidence matrix $\mathbf{B}$ and Laplacian matrix $\mathbf{L}$. Prove that $\mathbf{L} = \mathbf{B}^T\mathbf{B}$. Now prove that $\mathbf{L}\mathbf{e} = 0$ using the fact that $\mathbf{L} = \mathbf{B}^T\mathbf{B}$.

3 Network Analysis Fundamentals

3.1 Eigen-fundamentals

Eigen-information of graph-related matrices turns out to be a powerful tool in analyzing structural information of graphs. For example, we will see that the second smallest eigenvalue of the Laplacian matrix of a graph gives information about the best cut we can find in the graph. In a broader setting, the eigen-information of a matrix provides powerful tools for faster matrix operations and can even be used for data compression.

Let $\mathbf{M}$ be any $n \times n$ matrix with entries over the real numbers $\mathbb{R}$. A vector $\mathbf{v}$ is an *eigenvector* of $\mathbf{M}$ with *eigenvalue* λ iff $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$. We call the tuple $(\mathbf{v}, \lambda)$ an *eigen-pair*. Suppose the matrix $\mathbf{M}$ has n eigenvectors $\mathbf{v}_j$ with eigenvalues λ_j. Set $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$ and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_n)$. Then we have that the equation

$$\mathbf{M}\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda}$$

holds column-wise by definition of the columns $\mathbf{v}_j$ being eigenvectors of $\mathbf{M}$ (verify this). Note that the matrix $\mathbf{V}$ is invertible iff the n eigenvectors $\mathbf{v}_j$ are linearly independent; if they are, we can then rearrange the above equation to yield the *diagonalization* of matrix $\mathbf{M}$, also called the *spectral decomposition* or *eigen decomposition* of matrix $\mathbf{M}$:

$$\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}. \tag{1}$$

If a matrix $\mathbf{M}$ has a diagonalization, we say it is *diagonalizable*. When discussing eigenvalues of a matrix, we give the following conventional ordering to the eigenvalues: $\mathbf{M}$ has eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots, \geq \lambda_{n-1} \geq \lambda_n.$$

The following theorem guarantees that adjacency matrices of undirected graphs always have a nice diagonalization:

Theorem 1. *Let $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ be a symmetric. Then $\boldsymbol{M}$ has n linearly independent eigenvectors $\boldsymbol{v}_j$, and so we can express $\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$.*

Since undirected graphs (and, in particular, standard graphs) have symmetric adjacency matrices $\mathbf{A}$, we can always make use of the diagonalization of $\mathbf{A}$ and $\mathbf{L}$.

Recall that the *rank* of a matrix is equal to (1) the dimension of the columnspace of $\mathbf{M}$, (2) the dimension of the rowspace of $\mathbf{M}$.

Proposition 1. *If $\boldsymbol{M}$ is a diagonalizable, then* $\text{rank}(\boldsymbol{M})$ *is equal to the number of nonzero eigenvalues $\lambda_j \neq 0$ of $\boldsymbol{M}$, which equals the number of linearly independent eigenvectors $\boldsymbol{v}_j$ of $\boldsymbol{M}$.*

Proposition 2. *An $n \times n$ matrix $\boldsymbol{M}$ is invertible iff* $\text{rank}(\boldsymbol{M}) = n$.

An eigenvalue λ of a matrix $\mathbf{M}$ can have multiple linearly independent eigenvectors associated with it. The most straightforward example is the $n \times n$ identity matrix $\mathbf{I}$, which has the eigenvalue 1 with n different linearly independent eigenvectors; for exaxmple, each standard basis vector $\mathbf{e}_j$ is an eigenvector of $\mathbf{I}$, since $\mathbf{e}_j = \mathbf{I}\mathbf{e}_j$. If an eigenvalue λ of $\mathbf{M}$ has multiple lienarly independent eigenvectors, then those eigenvectors generate a subspace, called an *eigenspace* associated with the eigenvalue λ. For Egienvalue λ_j, let E_j denote the subspace of all eigenvectors $\mathbf{v}$ with eigenvalue λ_j, i.e. all vectors $\mathbf{v}$ such that $\mathbf{M}\mathbf{v} = \lambda_j\mathbf{v}$. Then the dimension of the subspace E_j is defined to be the *geometric multiplicity of the eigenvalue λ_j*. If a matrix has 0 as an eigenvalue, this is so important that we give a special name to the multiplicity of the eigenvalue 0 – if a matrix $\mathbf{M}$ has eigenvalue 0 with geometric multiplicity m_0, then we say $\mathbf{M}$ has *nullity* m_0, which we will denote nullity($\mathbf{M}$). This brings us to the "Rank-nullity theorem", which has a useful connection to Proposition 2:

Theorem 2 (Rank-nullity)**.** *For any $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ the rank and nullity of $\boldsymbol{M}$ satisfy* $\text{rank}(\boldsymbol{M}) = n - \text{nullity}(\boldsymbol{M})$. *In particular, $\boldsymbol{M}$ is invertible iff* $\text{nullity}(\boldsymbol{M}) = 0$, *i.e. 0 is not an eigenvalue of $\boldsymbol{M}$.*

Finally we connect these concepts to our analysis of networks, by looking more closely at Exercise 2.5 from the previous section. First, we restate the exercise as a result:

Proposition 3. *Let $\boldsymbol{L}$ be the Laplacian of a standard graph. Then $\boldsymbol{L}$ has eigenvalue $\lambda = 0$ with multiplicity at least 1, and $\boldsymbol{e}$ as an eigenvector: $\boldsymbol{L}\boldsymbol{e} = 0\boldsymbol{e}$.*

This means Laplacian matrices are never invertible, since they always have nullity at least 1. However, something neat happens if we look more closely at this result. But we'll get there – first, we need one more preliminary result.

Lemma 1. *Let G be a standard graph with k connected components, $G_1, \cdots, G_k$, with connected component G_j having Laplacian matrix $\boldsymbol{L}_j$. Then the graph G can have its nodes ordered so that the Laplacian matrix $\boldsymbol{L}$ of G is*

$$\boldsymbol{L} = \text{diag}(\boldsymbol{L}_1, \cdots, \boldsymbol{L}_k) = \begin{bmatrix} \boldsymbol{L}_1 & & \\ & \ddots & \\ & & \boldsymbol{L}_k \end{bmatrix}.$$

This lemma makes it easier to prove the following theorem, which gives us a first glimpse of how Linear Algebra can provide information useful to Graph Theory:

Theorem 3. *Let G be a standard graph with k connected components, and with Laplacian matrix $\boldsymbol{L}$. Then $k = \text{nullity}(\boldsymbol{L})$.*

In words, the multiplicity of zero as an eigenvalue of G's Laplacian matrix is the same as the number of distinct connected components of G.

3.2 Graph walks, graph matrices

Here we look at what I call the Fundamental Lemma of Network Analysis because of how often it is used.

Lemma 2 (Fundamental Lemma of Network Analysis)**.** *Let the graph G have adjacency matrix $\boldsymbol{A}$. Then the number of length k walks from node i to node j is equal to $(\boldsymbol{A}^k)_{ij}$.*

This lemma is so often used because it gives us a way to measure connectivity between different nodes – it gives us a way to measure, for a fixed node s, what nodes are most important to, or similar to, the node s? Consider the polynomial $p(x) = \sum_{k=1}^{N} x^k$. Then the vector

$$\mathbf{f} = p(\mathbf{A})\mathbf{e}_s = \sum_{k=1}^{N} (\mathbf{A}^k)\mathbf{e}_s$$

tells us for each node j in the graph, how many total walks are there from node s to node j of length $\leq N$? Large entries in the vector $\mathbf{f}$ end up being good predictors for nodes that are important to node s.

3.3 Exercises

3.1 Prove Lemma 2. Hint: begin with $\mathbf{A}^1$ as the base case, and proceed by induction. For the case $(\mathbf{A}^k)_{ij}$, think about the question "how do you make a walk from node i to node j that is of length k?".

4 More on structured sparsity

Here we begin our forray into notions of structured sparsity. We start with one of the simpler notions, that of the *degeneracy* of a graph G, which the Theory in Practice Lab has described as follows: "No good notion of sparsity *doesn't* imply bounded degeneracy." In other words, any good idea of G having structured sparsity should imply that G has bounded degeneracy. Next, we explore exactly what that means.

4.1 Degeneracy and k-cores

As usual, all graphs we consider here are standard graphs. A *k-core* is a graph in which every node had degree at least k. A k-core of a graph G is any induced subgraph of G that is also a k-core. The *k-core number of a node v* in G is the largest number k for which v is contained in a k-core of G. The *k-core of G* is the largest number k for which G has a k-core; the k-core number of G is also called the *degeneracy* of G.

4.1.1 Uses and Examples

Degeneracy provides a much more useful notion of "density" than, say, maximum degree or average degree. For example, a star graph on n nodes is a very sparse graph, yet it has maximum degree $n-1$, which "sounds like it might be dense". In contrast, the degeneracy of a star is always 1, which reveals its true sparsity. Similarly, if a graph has constant average degree it does not guarantee any useful bound on the maximum clique size: an n node graph with kn edges can have a clique of size $\sim \sqrt{(k-1)n}$ (by constructing a path graph on n nodes, then putting all other edges in a clique). In contrast, the degeneracy of such a graph would be lower-bounded by the clique size, because a k-clique is always a $(k-1)$-core.

We could also think about this from the other direction, i.e. that the core number (degeneracy) of G always gives a bound on the max clique number. These bounds are not always useful, though: there are graphs with arbitrarily large gaps between the max-clique size and k-core number. For example, a hypercube on 2^n nodes is always an n-core, but the largest clique is always of size 2.

Another useful feature related to degeneracy and core numbers is the idea of a *degeneracy ordering* of a graph's nodes. This is any ordering on the nodes of G that sorts the nodes, in ascending order, according to the nodes' k-core numbers. It is called the degeneracy ordering because if you delete these nodes, one-by-one, according to this ordering, then the core number of node v is exactly equal to the degree of v at the time v is deleted. A degeneracy ordering can be useful for processing the nodes and edges of a graph in an efficient way.

For a linear-time algorithm that computes the k-core numbers of all nodes in G, see [1]. To use k-cores to accelerate computation of the max-clique in a graph, see [4].

4.1.2 Exercises

4.1.1 If G has maximum degree d, prove that degeneracy of G is bounded above by d.

4.1.2 If G has degeneracy k, does this imply any upper bound on the number n of nodes in the graph? For a fixed number of nodes n, does degeneracy k imply an upper bound on the number of edges?

4.1.3 If G has degeneracy k, does this imply any lower bound on the number n of nodes in the graph? For a fixed number of nodes n, does degeneracy k imply a lower bound on the number of edges?

4.2 Tree-decompositions and tree-width

Definition We will, eventually, give a formal definition of a tree-decomposition of a graph G, but first we want to approach the topic intuitively. A tree-decomposition is not "a decomposition of a graph into trees" (this idea is closer to the *arboricity* of a graph). Rather, a tree-decomposition is a map from the nodes of G to some tree T such that each node of T (which we call a "bag" and usually denote by X_j) contains a set of

vertices of G, and we can think of T as being a map of G that tells us "how to process the nodes of G in an efficient order". Let's take the simplest possible type of tree for T, a path. If we have a tree-decomposition of G such that its nodes are all mapped into bags of T that are lined up in a path $(X_1, X_2, \cdots, X_k)$, then to solve a problem on the entire graph G it can suffice to simply perform computations on the nodes of G one bag X_j at a time before moving on to the next bag X_{j+1}. To show why the tree-decomposition can be useful as a "processing map", we next describe it more rigorously.

Formally, a *tree-decomposition* of a graph $G = (V, E)$ consists of a tree $T = (V_T, E_T)$ whose nodes we call "bags", such that:

T.1 Every node of G is contained in at least one bag X_j of T.

T.2 Every edge of G has both its endpoints contained in at least one bag X_j of T.

T.3 For each node v in G, the set of bags $X_j \in V_T$ containing v induces a connected subtree of T.

Any such mapping from G to a tree T that satisfies these properties is a tree-decomposition of G; a given graph can have many tree-decompositions. For a given tree decomposition, we define the *width* of the decomposition to be $\max_{X_j \in V_T}(|X_j| - 1)$. The *tree-width* of the graph G, $\text{tw}(G)$, is then the minimum width of all tree-decompositions of G.

Understanding its usefulness The basic idea of the usefulness of a tree decomposition is that the entire graph can be examined by simply examining all nodes in each bag, one bag at a time. If the graph has low tree-width, then no big is too large, and so each bag can be swiftly processed. Thus, "graphs with bounded tree-width" is a class of structurally sparse graphs for which there exist a lot of fixed-parameter tractable algorithms, where the parameter that gets fixed is the tree-width of the graph, $\text{tr}(G) = k$.

Tree-decompositions also have connections to vertex seperators. A *vertex separator* of a connected graph $G = (V, E)$ is a set $S \subset V$ of nodes such that deleting $G - S$ is disconnected. Vertex separators can be very useful in divide-and-conquer style algorithms, and in distributing graphs across multiuple processors for more efficient computation or storage. It turns out that each bag X_j in any tree decomposition of G is a vertex separator.

An *edge separator* of a connected graph $G = (V, E)$ is a set $F \subset E$ of edges such that $G - F$ is disconnected. This is also called an *edge cut* or simply a *cut*. A cut, or edge separator F, is "balanced" if deleting F leaves behind two graph components that are not too different in size (this is a hand-wavey, intuitive term, not a rigorous once). To be more precise, we define the *conductance* of a cut as follows. First note that any cut F necessarily splits the set of nodes into two components, S and $G - S$. So, we can define a cut either by its edge set, F, or, equivalently, by the set of nodes that it disconnects, $F = \text{cut}(S, G - S)$. Recalling that $\text{vol}(S)$ is defined to be the sum of the degrees of the nodes in S, $\text{vol}(S) = \sum_{v \in S} d(v)$, we can now define the condutance of the cut (edge separator) $F = \text{cut}(S, G - S)$ to be

$$\text{conductance}(S) = \frac{\text{cut}(S, G - S)}{\min\{\text{vol}(S), \text{vol}(G - S)\}}, \tag{2}$$

which is conventionally denoted $\phi(S)$. Note that condutance is always between 0 and 1; when $\phi(S) = 0$, the cut is very small because S is not connected to the rest of G (so the cut contains zero edges), whereas when $\phi(S) = 1$ the cut is larger, because all edges connected to nodes in S leave the set S and have their other endpoints in the set $G - S$ (so the cut contains a "maximal" number of edges).

The connection of topics here is that every cut (edge separator) containing t edges guarantees the existence of a vertex separator that contains no more than t nodes. To see this, note that a cut $F = \text{cut}(S, G - S)$ containing t edges is incident to no more than t nodes in S (and no more than t nodes in $G - S$). Thus, deleting those $s \leq t$ nodes will separate the graph into two pieces, and so some subset of the nodes in S is a vertex separator.

Proposition 4. *Every cut (edge separator) in G yields at least one vertex separator.*

This provides a direct connection from conductance and cuts to vertex separators and, hence, tree-decompositions. Thus, we next move our focus to eigenvalue approaches to understanding conductance and cuts in a graph. Keep in mind that the benefit of tree decompositions is not the size of the tree T, but the width of the bags associated with the tree – in other words, a tree decomposition is most helpful if all of the vertex separators it provides are small separators (contain a small number of nodes).

4.2.1 Exercises

4.2.1 Let G be a connected graph with a node v of degree 1. Prove that deleting v from G does not disconnect G; i.e. prove that $(G - v)$ is connected.

4.2.2 Prove that a tree must have at least one node of degree 1. Then use that result to prove that a tree must have at least two nodes of degree 1.

4.2.3 Prove that a connected graph in which every node has degree at least 2 must contain at least one cycle.

4.2.4 Prove that every path graph P has a tree-decomposition of tree-width 1. (Hint: take the path on n nodes and tell how to construct construct the bags. (Sub-hint: width 1 means that each bag has exactly two nodes in it.))

4.2.5 Prove that if G is a tree, then its tree-width is 1. (Hint: tree-width 1 means that the tree decomposition has width 1, which means every bag in that decomposition contains no more than 2 nodes.) (Hint 2: make a couple small trees and explicitly construct tree-decompositions for them with width 1. Notice anything about how each edge of the tree gets mapped to bags in the tree-decomposition?)

4.2.6 Let $G = (V, E)$ be a connected graph. Let G have a tree decomposition with tree $T = (V_T, E_T)$ (i.e. V_T is the set of bags of the tree-decomposition). Fix any bag $X_j \in V_T$ and recall that a bag in a tree-decomposition consists of a set of nodes in the original graph, G. Let G' be the graph that results from deleting from G all nodes v in the bag X_j, i.e. let $G' = G - X_j$. Prove that G' is disconnected. (i.e. prove that each bag of a tree-decomposition for a graph G gives rise to a vertex separator of G.)

4.2.7 Prove that rule T.3 in the definition of tree-decomposition is equivalent to the following: "for any two bags X_i, X_j in the tree T, every bag X on the path in T connecting X_i to X_j must contain their intersection, i.e. $X_i \cap X_j \subseteq X$." (i.e. show that we could replace rule T.3 with this rule, and the resulting definition of tree-decomposition would be equivalent).

This section contains exploring notions of structured sparsity. We will further develop some ideas useful to the study of tree-decompositions and degeneracy.

4.3 Minor subgraphs

As usual, all graphs we consider here are standard graphs. Recall that a graph G contains H as a *minor subgraph* (also called simply "a minor") if you can obtain a copy of H by a sequence of node deletions, edge deletions, and edge contractions (defined below). We will later describe important variations on this idea (e.g. "shallow" and "topological" minor subgraphs), but first we want to make clear the importance of subgraph minors to the study of sparsity.

We will illustrate the connection of minors to sparsity with an example. One canonical example of "sparse" graphs are the family of planar graphs – planar graphs have $O(n)$ edges and bounded degeneracy. It turns out that planar graphs are completely characterized by subgraph minors. To elaborate: a graph G is planar if and only if G does not contains K_5 or $K_{3,3}$ as minors (where K_5 is the complete graph on 5 nodes, and $K_{3,3}$ is the complete-bipartite graph with 3 nodes in each partition). This is an example of a type of sparsity that is completely characterized by specific graph minors.

The point here is that having knowledge of the minors of a graph can help determine the sparsity of the graph. Therefore, we would like an easy way to learn some information about the minors of a graph. We are not aiming to characterize any graph family completely (that is a little ambitious); rather, our goal is to try to reveal connections between a graph's eigenproperties and a graph's minors.

4.3.1 Terminology

Operations To perform an *edge contraction* on an edge with endpoints v_1, v_2, create a new node v, attached an edge from v to each neighbor of v_1 and each neighbor of v_2, then delete nodes v_1 and v_2. You can think of this as merging the two nodes v_1, v_2 into each other.

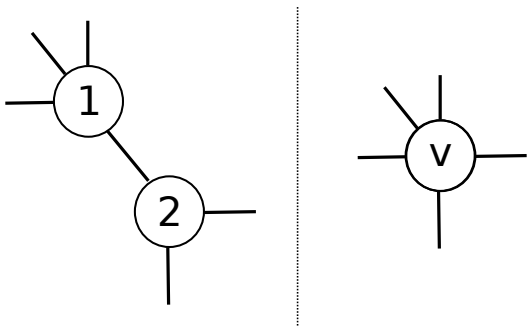

Figure 1: (*Left.*) Nodes v_1 and v_2 connected by an edge. (*Right.*) The graph after contracting edge v_1v_2, merging nodes v_1 and v_2 into one new node, v.

A *subdivision* (or "edge subdivision") is sort of the reverse of an edge contraction: given an edge with endpoints v_1, v_2, to subdivide the edge e_{12}, create a new node v, connect v with an edge to node v_1 and an edge to v_2, then delete edge e_{12}. This operation is like spitting edge e_{12} into two pieces connected by a new node v. Note that a subdivision introduces a node of degree two. The reverse process, "un-subdividing", is called *smoothing* or *dissolving* a node of degree two. If node v has degree two and is connected to nodes v_1 and v_2, then smoothing node v is performed by connecting node v_1 and v_2 with an edge, then deleting node v. See Figure 2 for an illustration. Smoothing a node of degree two can also be thought of as an edge-contraction on either of the edges touching the node.

A graph T is a *topological minor* of G if you can produce G from T by subdividing edges of T, and adding nodes and edges to T. Put another way, T is a topological minor ("top-minor" for short) of G if you

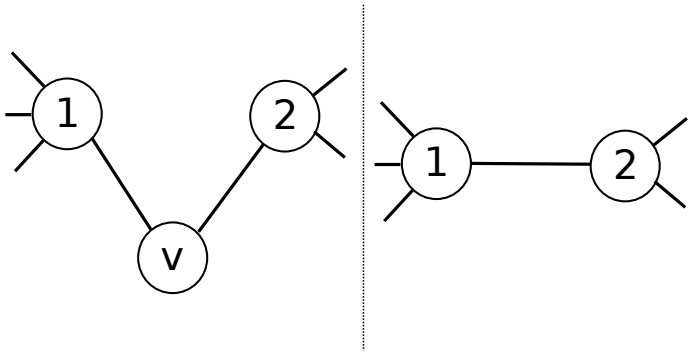

Figure 2: (*Left.*) Nodes v_1 and v_2 separated by a node of degree two, v. (*Right.*) The graph after node v has been smoothed; alternatively, this can be understood as contracting edge v_1v or edge v_2v. Note that viewing the figure from right to left gives an illustration of the edge between v_1 and v_2 being subdivided.

can produce T from G by deleting nodes and edges, and smoothing nodes (of degree two). Note that all topological minor subgraphs are also minor subgraphs. See Figure 3 for an example of a graph with a K_4 as a topological minor. For contrast, Figure 4 displays a graph that does not contain K_5 as a subgraph or as a topological minor subgraph (no node has degree 2, so no nodes can be smoothed), but it does contain K_5 as a minor subgraph.

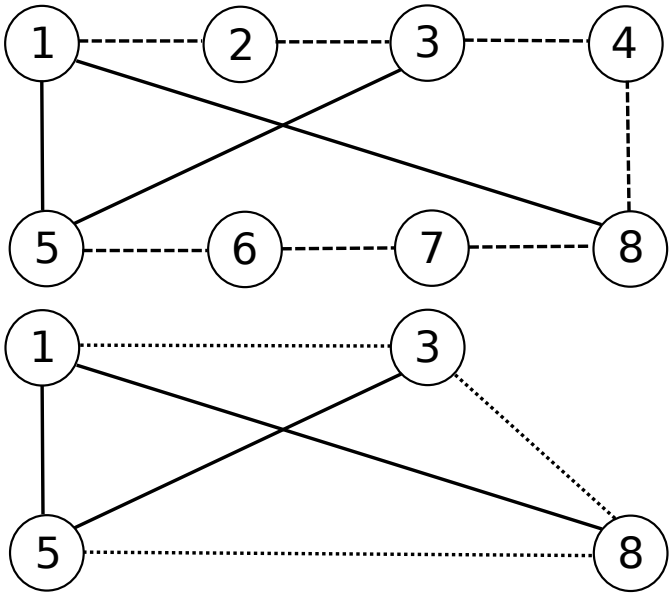

Figure 3: In the top graph, smoothing the nodes connected to only dashed egdes yields the graph on bottom as a topological minor. Alternatively, this can be thought of as contracting the dashed edges. Note that the dotted edges in the bottom graph indicate the edges produced by the smoothings in the top graph. Finally, viewing this figure from bottom to top, this illustrates an instance of K_4 where we then perform several subdivisions on the dotted edges to produce the graph on top.

object	operations allowed
minor subgraph	node deletions
	edge deletions
	edge contractions
topological minor subgraph	node deletions
	edge deletions
	node smoothings (contracting edges touching a node of degree 2)

5 Eigen-fundamentals, Part 2

Don't worry about this section for a while – this section will come in handy when we start trying to *prove* things about eigenvalues of certain graphs.

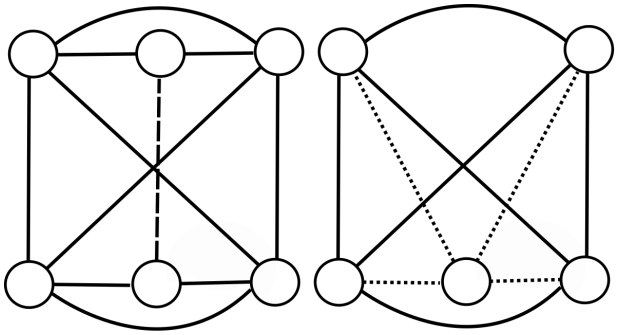

Figure 4: (*Left.*) This graph does not contain K_5 as a subgraph, or as a topological minor subgraph. (*Right.*) Contracting the dashed edge on the left yields a new node, with new dotted edges in the graph on the right. The graph on the right is K_5, so this proves the graph on the left has K_5 as a minor subgraph.

Here we describe some connections of eigenvalue theory to graph cuts and conductance. Recall that the Laplacian matrix is defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, and that it satisfifes $\mathbf{L} = \mathbf{B}^T\mathbf{B}$, where $\mathbf{B}$ is the edge-node-incidence matrix. Then we define the *normalized Laplacian matrix* to be $\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, and we denote its eigenvalues by

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1} \geq \lambda_n.$$

Defining the *normalized adjacency matrix* to be $\hat{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, note that the normalized Laplacian is simply $\mathbf{I} - \hat{\mathbf{A}}$.

In this section we will build up to proving a connection of the second smallest eigenvalue of $\hat{\mathbf{L}}$ to the best (smallest) conductance cut in the graph G. We begin by proving a simple but useful fact about the eigenvalues of $\mathbf{L}$ and $\hat{\mathbf{L}}$.

A $\mathbf{M} \in \mathbb{R}^{n\times n}$ with all real eigenvalues is *positive definite* iff all of its eigenvalues are positive. Instead, we say $\mathbf{M}$ is *postive semi-definite* iff all of its eigenvalues are nonnegative. (Note that all positive definite matrices are positive semi-definite, but a positive semi-definite matrix (which might have eigenvalues equal to 0) is not necessarily positive definite.)

Note that these definitions are restricted to matrices with all real eigenvalues – the definitions can be extended to matrices with possibly complex eigenvalues, but this is beyond the scope of this project. (This is because all matrices we are concerned with are similar to symmetric matrices, which are guaranteed to have all real-valued eigenvalues, see Theorem 6 below.)

Speaking of symmetric matrices, a particularly well-studied area within positive definite and semi-definite matrices is the subject of symmetric postive definite (SPD) and semi-definite (SPSD) matrices. So we will have the luxury of dealing in SPD and SPSD matrices here, which enjoy some useful properties that more general postiive (semi-)definite matrices do not. For example:

Theorem 4. *Let $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ be symmetric. Then $\boldsymbol{M}$ is SPSD iff there exists some $\boldsymbol{U} \in \mathbb{R}^{n\times r}$ such that $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{U}^T$.*

Proof. We will prove only the easier direction of this proposition here. We prove the other direction below, after establishing a little more machinery. Assume that $\mathbf{M} = \mathbf{U}\mathbf{U}^T$, and let $(\mathbf{v}, \lambda)$ be any eigen-pair of the matrix $\mathbf{M}$, and let $\mathbf{v}$ be a unit eigen-vector. Then we have

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v} \qquad \text{definition of eigen-vector} \tag{3}$$

$$\mathbf{v}^T\mathbf{M}\mathbf{v} = \lambda\mathbf{v}^T\mathbf{v} \qquad \text{left-multiply by } \mathbf{v}^T \tag{4}$$

$$\mathbf{v}^T\mathbf{U}\mathbf{U}^T\mathbf{v} = \lambda \qquad \text{substitute } \mathbf{M} = \mathbf{U}\mathbf{U}^T \text{ and simplify} \tag{5}$$

$$(\mathbf{U}^T\mathbf{v})^T(\mathbf{U}^T\mathbf{v}) = \lambda \tag{6}$$

and since $(\mathbf{U}^T\mathbf{v})^T(\mathbf{U}^T\mathbf{v}) \geq 0$, we know that $\lambda \geq 0$, proving that $\mathbf{M}$ is SPSD. This trick will come back to help us a number of times, so keep it in your pocket. □

Recalling that $\mathbf{L} = \mathbf{B}^T\mathbf{B}$, this proposition implies that the Laplacian matrix $\mathbf{L}$ is positive semi-definite. We also know $\mathbf{L}$ is symmetric, so $\mathbf{L}$ is SPSD. Is the Laplacian positive definite? (i.e. not semi-definite?) No: we showed in an earlier section that $\mathbf{L}\mathbf{e} = 0\mathbf{e}$, so we definitely know that $\mathbf{L}$ is semi-definite, not definite. But what about $\hat{\mathbf{L}}$? We know it is still symmetric, from its definition. Furthermore, $\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{B}^T\mathbf{B}\mathbf{D}^{-1/2} = \hat{\mathbf{B}}^T\hat{\mathbf{B}}$, where $\hat{\mathbf{B}} = \mathbf{B}\mathbf{D}^{-1/2}$. Thus, $\hat{\mathbf{L}}$ is also SPSD, with nullvector $\mathbf{v}_n = (\mathbf{D}^{1/2}\mathbf{e})$:

$$\hat{\mathbf{L}}(\mathbf{D}^{1/2}\mathbf{e}) = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\mathbf{e} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{e} = 0\mathbf{e}.$$

With this we have compeleted a proof of the following.

Proposition 5. *For any standard graph, $\boldsymbol{L}$ and $\hat{\boldsymbol{L}}$ are SPSD with null-vectors $\boldsymbol{e}$ and $\boldsymbol{D}^{1/2}\boldsymbol{e}$, respectively.*

We will focus for now on the matrix $\hat{\mathbf{L}}$. Denote its eigenpairs by $(\mathbf{v}_j, \lambda_j)$, and recall that we just finished proving $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1} \geq \lambda_n = 0$. To continue, we need the following essential result about the eigen-properties of symmetric matrices.

Proposition 6. *Let $\boldsymbol{M}$ be any real, $n \times n$, symmetric matrix. Then not only is $\boldsymbol{M}$ diagonalizable (c.f. Theorem 1), i.e. $\boldsymbol{M} = \boldsymbol{V\Lambda V}^{-1}$, but we also have that $\boldsymbol{M}$ has real eigenvalues and all its eigenvectors are orthogonal, so $\boldsymbol{M} = \boldsymbol{V\Lambda V}^T$ and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$.*

Proof. We are going to assume Theorem 1, i.e. assume that symmetric $\mathbf{M}$ has diagonal form $\mathbf{M} = \mathbf{V\Lambda V}^{-1}$. Let $\mathbf{v}_i$ and $\mathbf{v}_j$ be eigenvectors corresponding to distinct eigenvalues. Claim: then $\mathbf{v}_i^T\mathbf{v}_j = 0$. To prove this, note that

$$\begin{aligned}
\lambda_i \mathbf{v}_i^T \mathbf{v}_j &= (\lambda_i \mathbf{v}_i^T)\mathbf{v}_j && (7)\\
&= (\mathbf{v}_i^T \mathbf{M}^T)\mathbf{v}_j && (8)\\
&= \mathbf{v}_i^T(\mathbf{M}^T \mathbf{v}_j) && (9)\\
&= \mathbf{v}_i^T(\mathbf{M}\mathbf{v}_j) \qquad \text{because } \mathbf{M} = \mathbf{M}^T && (10)\\
&= \lambda_j \mathbf{v}_i^T \mathbf{v}_j && (11)
\end{aligned}$$

so that $\lambda_i \mathbf{v}_i^T\mathbf{v}_j = \lambda_j \mathbf{v}_i^T\mathbf{v}_j$. If $\mathbf{v}_i^T\mathbf{v}_j \neq 0$, then we can divide both sides by $\mathbf{v}_i^T\mathbf{v}_j$ to yield $\lambda_i = \lambda_j$, but this would contradict the assumption that the eigenvalues were distinct. Thus, $\mathbf{v}_i^T\mathbf{v}_j = 0$, proving the claim. Observe that this part of the proof used the fact that $\mathbf{M} = \mathbf{M}^T$.

We have proved that eigenvectors of distinct eigenvalues are orthogonal. It remains to show that eigenvectors of the same eigenvalue are orthogonal. For any eigenvalue, λ_j, let E_j be the space spanned by all eigenvectors of $\mathbf{M}$ associated to λ_j. So for all $\mathbf{u}$ such that $\mathbf{Mu} = \lambda_j\mathbf{u}$, we have $\mathbf{u} \in E_j$; this is called the *eigenspace* associated to the eigenvalue λ_j. Note that it is in fact a vector space: given any $\mathbf{a}, \mathbf{b} \in E_j$ and any scalar γ we have $\mathbf{M}(\mathbf{a} + \gamma\mathbf{b}) = \mathbf{Ma} + \gamma\mathbf{Mb} = \lambda_j\mathbf{a} + \gamma\lambda_j\mathbf{b} = \lambda(\mathbf{a} + \gamma\mathbf{b}) \in E_j$. This proves E_j is in fact a subspace. By standard linear algebra facts, every subspace of a finite dimensioned vectorspace (in this case $\mathbb{R}^n$) has an orthogonal basis. Let $\{\mathbf{b}_1, \cdots, \mathbf{b}_d\}$ be an orthogonal basis for the eigenspace E_j, and set $\mathbf{V}_j = [\mathbf{b}_1, \cdots, \mathbf{b}_d]$. Then $\mathbf{V}_j^T\mathbf{V} = \mathbf{I}$ by construction. Furthermore, because $\mathbf{V}_i$ and $\mathbf{V}_j$ are bases for the eigenspaces of distinct eigenvalues, we know that $\mathbf{V}_i$ and $\mathbf{V}_j$ are orthogonal: $\mathbf{V}_i^T\mathbf{V}_j = 0$. Thus, setting $\mathbf{V} = [\mathbf{V}_1, \cdots, \mathbf{V}_g]$ we have $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ by construction. This completes the proof. □

Remark. Now that we have this theorem, we can complete a proof of an earlier theorem. By Theorem 6 we know we can write $\mathbf{L} = \mathbf{V\Lambda V}^T$. Since $\mathbf{L}$ is SPSD, we also know that each eigenvalue is nonnegative, and so we can take the square root of each eigenvalue, and write $\mathbf{\Lambda} = \mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}$. Since $\mathbf{\Lambda}$ is diagonal, we have $\mathbf{\Lambda}^{1/2} = (\mathbf{\Lambda}^{1/2})^T$. Thus, we can write $\mathbf{L} = \mathbf{V\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}(\mathbf{V\Lambda}^{1/2})^T$. Setting $\hat{\mathbf{V}} = \mathbf{V\Lambda}^{1/2}$ we can write $\mathbf{L} = \hat{\mathbf{V}}\hat{\mathbf{V}}^T$. This provides a constructive proof that symmetric positive semi-definite matrices can be written $\mathbf{M} = \mathbf{UU}^T$, completing the proof of Theorem 4.

5.0.1 Rayleigh quotients and minimax

We need just a few more eigen-tools before we can analyze graph spectra. Consider again the equation we used earlier $\mathbf{v}^T\mathbf{Mv}$. For any eigenvector $\mathbf{v}$ that has unit length (i.e. $\mathbf{v}^T\mathbf{v} = 1$), we know that $\mathbf{v}^T\mathbf{Mv} = \lambda$. If $\mathbf{v}$ does not have unit length, then instead we get $\mathbf{v}^T\mathbf{Mv} = \lambda\mathbf{v}^T\mathbf{v}$. If we divide this equation by $\mathbf{v}^T\mathbf{v}$, we get what is called a *Rayleigh quotient* with respect to $\mathbf{M}$ and $\mathbf{v}$: $R(\mathbf{M}, \mathbf{v}) = \frac{\mathbf{v}^T\mathbf{Mv}}{\mathbf{v}^T\mathbf{v}}$. Rayleigh quotients are useful tools for analyzing eigenvalues and eigenvectors, because we know that for any eigenpair $(\mathbf{v}, \lambda)$ the Rayleigh quotient always satisfies $R(\mathbf{M}, \mathbf{v}) = \lambda$.

It turns out that there is a way to define the eigenvectors and values using these Rayleigh quotients.

Proposition 7. *For any square, real symmetric matrix $\boldsymbol{M}$, the smallest eigenvalue λ_n can be defined as follows:*

$$\lambda_n := \min_{\boldsymbol{x} \neq 0 \in \mathbb{R}^n} \frac{\boldsymbol{x}^T\boldsymbol{Mx}}{\boldsymbol{x}^T\boldsymbol{x}}. \tag{12}$$

Proof. To see that this is the case, first note that $R(\mathbf{M}, \mathbf{v}_n) = \mathbf{v}_n^T\mathbf{M}\mathbf{v}_n/\mathbf{v}_n^T\mathbf{v}_n = \lambda_n$, so certainly the minimum in Equation (12) is at least as small as λ_n. Now suppose that there were a vector $\mathbf{x}$ and a value $\gamma \in \mathbb{R}$ such that $\mathbf{x}^T\mathbf{Mx}/\mathbf{x}^T\mathbf{x} = \gamma < \lambda_n$. Since $\mathbf{M}$ is real and symmetric, we know that it has an eigen-decomposition $\mathbf{M} = \mathbf{V\Lambda V}^T$. Thus we can write

$$\begin{aligned}
\gamma &= \mathbf{x}^T\mathbf{V\Lambda V}^T\mathbf{x}/\mathbf{x}^T\mathbf{Ix} && (13)\\
&= (\mathbf{V}^T\mathbf{x})\mathbf{\Lambda}(\mathbf{V}^T\mathbf{x})/\mathbf{x}^T\mathbf{VV}^T\mathbf{x} && (14)\\
&= (\mathbf{V}^T\mathbf{x})\mathbf{\Lambda}(\mathbf{V}^T\mathbf{x})/(\mathbf{V}^T\mathbf{x})^T(\mathbf{V}^T\mathbf{x}). && (15)
\end{aligned}$$

For simplicity of notation, let $\mathbf{y} = \mathbf{V}^T\mathbf{x}$. This expression can be re-written as $\gamma = \mathbf{y}^T\mathbf{\Lambda y}/\mathbf{y}^T\mathbf{y}$, where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. Writing it out explicitly, we have

$$\begin{aligned}
\gamma &= \mathbf{y}^T\mathbf{\Lambda y}/\mathbf{y}^T\mathbf{y} && (16)\\
&= \sum_{y_j^2\lambda_j} / \sum_{y_j^2}. && (17)
\end{aligned}$$

Thinking of γ as a function of the variables $y_1, \cdots, y_n$, we can apply basic calculus to the above expression to prove that the minimum value of $\gamma(\mathbf{y})$ is $\gamma = \lambda_n$. □

It further turns out that this characterization of the smallest eigenvalue can be extended to define the other eigenvalues as well. The full theorem lies beyond the scope of this project, so we state here a simplified version of the *min-max theorem* adapted to our specific circumstances.

Proposition 8. *Let $\boldsymbol{M}$ be a square, real symmetric matrix with smallest eigenpair $\lambda_n = 0$ and $\boldsymbol{v}_n$. Then λ_{n-1} is equal to*

$$\lambda_{n-1} = \min_{\boldsymbol{x}:\boldsymbol{x}^T \perp \boldsymbol{v}_n, \boldsymbol{x} \neq 0} \frac{\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}^T}{\boldsymbol{x}^T \boldsymbol{x}} \tag{18}$$

where $\boldsymbol{x}$ must be real-valued vectors.

We will not prove this.

Cheeger's Inequality At last we have established enough foundation to begin describing spectral characterizations of graph properties. Here we state a famous result relating the best conductance cut in a graph to the graph's eigenvalues. We've proved before that the normalized Laplacian matrix has $\lambda_n = 0$ as its smallest eigenvalue. Furthermore, it follows from the eigenvalue structure of $\mathbf{L}$ that the multiplicity of the eigenvalue $\lambda_n = 0$ is equal to the number of connected components in the graph. It is the case that the graph is connected if and only if the eigenvalue λ_{n-1} is nonzero – for this reason we call λ_{n-1} the *algebraic connectivity* of the graph. The algebraic connectivity is 0 when the graph is disconnected, and nonzero when the graph is connected – but here we present a theorem showing that the magnitude of λ_{n-1} actually tells us *how well connected* the graph is, i.e. how many edges do we need to cut to separate the graph into two roughly balanced pieces?

Theorem 5 (Cheeger's Inequality)**.** *Let G have normalized Laplacian $\hat{\boldsymbol{L}}$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_{n-1} \geq \lambda_n$. Let S be the smallest-conductance cut in G, and denote its conductance by $\Phi = \phi(S)$. Then we have*

$$\sqrt{2}\sqrt{\lambda_{n-1}} \geq \Phi \geq \tfrac{1}{2}\lambda_{n-1}$$

Before proving the easy direction in this theorem, we need a lemma.

Lemma 3. *Let $\boldsymbol{L}$ be the Laplacian matrix for any standard graph $G = (V, E)$. Then $\boldsymbol{f}^T \boldsymbol{L}\boldsymbol{f} = \sum_{e_{ij} \in E} (f_i - f_j)^2$, i.e. $\boldsymbol{f}^T \boldsymbol{L}\boldsymbol{f}$ equals the sum of the squares of the differences of $\boldsymbol{f}$ across each edge.*

The proof is left as an exercise.

Of Cheeger's Inequality. For a full proof, consult [2]. We present just the easier direction here, $\Phi \geq \frac{1}{2}\lambda_{n-1}$.

Recall that for $\hat{\mathbf{L}}$ we have that $\mathbf{v}_n = \mathbf{D}^{1/2}\mathbf{e}$ is the eigenvector of $\hat{\mathbf{L}}$ corresponding to the smallest eigenvalue $\lambda_n = 0$. So by Theorem 8 we can express the algebraic connectivity of the graph as

$$\lambda_{n-1} = \min_{\mathbf{x}:\mathbf{x}^T \perp \mathbf{D}^{1/2}\mathbf{e}, \mathbf{x} \neq 0} \frac{\mathbf{x}^T \hat{\mathbf{L}} \mathbf{x}^T}{\mathbf{x}^T \mathbf{x}} = \min_{\mathbf{x}:\mathbf{x}^T \perp \mathbf{D}^{1/2}\mathbf{e}, \mathbf{x} \neq 0} R(\hat{\mathbf{L}}, \mathbf{x})$$

We will show that conductance can be expressed as a Rayleigh quotient related to the above as follows. The conductance of a set S equals $\phi(S) = \text{cut}(S, G - S)/\min\{\text{vol}(S), \text{vol}(G - S)\}$. Let $\mathbf{e}_S$ be the indicator vector of the set S, i.e. $\mathbf{e}_S$ is 0 everywhere, except 1s in indices corresponding to nodes in the set S. Then $\text{vol}(S) = \mathbf{e}_S^{\mathbf{D}} \mathbf{e}_S$. For ease of notation, we will switch from using $G - S$ to using $S^c = G - S$ to denote the complement of a set of nodes. Define the vector $\mathbf{y} = \frac{1}{\text{vol}(S)}\mathbf{e}_S - \frac{1}{\text{vol}(S^c)}\mathbf{e}_{S^c}$. We claim that $\mathbf{x} = \mathbf{D}^{1/2}\mathbf{y}$ is orthogonal to $\mathbf{D}^{1/2}\mathbf{e}$, so that $\lambda_{n-1} \leq R(\hat{\mathbf{L}}, \mathbf{x})$ must be true by Theorem 8; and furthermore that $R(\hat{\mathbf{L}}, \mathbf{x})$ is related to the conductance Φ that we seek to bound.

Proof of the first claim:

$$\begin{aligned}
\mathbf{x}^T \mathbf{D}^{1/2}\mathbf{e} &= \mathbf{y}^T \mathbf{D}^{1/2}\mathbf{D}^{1/2}\mathbf{e} && (19)\\
&= \left(\tfrac{1}{\text{vol}(S)}\mathbf{e}_S - \tfrac{1}{\text{vol}(S^c)}\mathbf{e}_{S^c}\right)^T \mathbf{D}\mathbf{e} && (20)\\
&= \tfrac{1}{\text{vol}(S)}\mathbf{e}_S^T \mathbf{D}\mathbf{e} - \tfrac{1}{\text{vol}(S^c)}\mathbf{e}_{S^c}\mathbf{D}\mathbf{e} && (21)\\
&= 1 - 1, && (22)
\end{aligned}$$

as desired.

To prove the second claim, observe the following:

$$\begin{aligned}
\mathbf{y}^T \mathbf{L}\mathbf{y} &= \mathbf{y}\mathbf{D}^{1/2}\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\mathbf{y} && (23)\\
&= \mathbf{x}^T \hat{\mathbf{L}}\mathbf{x} && (24)
\end{aligned}$$

and $\mathbf{y}^T\mathbf{D}\mathbf{y} = \mathbf{x}^T\mathbf{x}$. These equivalencies enable us to write $R(\hat{\mathbf{L}}, \mathbf{x}) = \mathbf{y}^T\mathbf{L}\mathbf{y}/\mathbf{y}^T\mathbf{D}\mathbf{y}$, so we can prove something about $R(\hat{\mathbf{L}}, \mathbf{x})$ by instead working on the more convenient expression $\mathbf{y}\mathbf{L}\mathbf{y}/\mathbf{y}^T\mathbf{D}\mathbf{y}$.

Using Lemma 3 we can write $\mathbf{y}\mathbf{L}\mathbf{y} = \sum_{e_{ij} \in E}(y_i - y_j)^2$. From the definition of $\mathbf{y}$, we know that if i and j are both in the same set (S or S^c), then $(y_i - y_j)^2 = 0$; if instead i and j are in opposite sets, we have

$(y_i - y_j)^2 = (\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)})^2$ regardless of which node is on which side. This occurs only when the nodes are the endpoints of an edge e_{ij} crossing the cut $\text{cut}(S, S^c)$. Thus, we have

$$\mathbf{yLy} = \sum_{e_{ij} \in \text{cut}(S)} (\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)})^2 \tag{25}$$

$$= (\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)})^2 \sum_{e_{ij} \in \text{cut}(S)} \tag{26}$$

$$= (\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)})^2 \,\text{cut}(S) \tag{27}$$

and hence, $\mathbf{y}^T\mathbf{Ly} = (\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)})^2 \,\text{cut}(S)$. At the same time, we have $\mathbf{y}^T\mathbf{Dy} = \sum_{k=1}^n y_k^2 d(k) = \sum_{v \in S} \frac{1}{\text{vol}(S)}^2 d(v) + \sum_{u \in S^c} \left(\frac{-1}{\text{vol}(S^c)}\right)^2 d(u)$. Factoring yields $\frac{1}{\text{vol}(S)}^2 \sum_{v \in S} d(v) + \frac{1}{\text{vol}(S^c)}^2 \sum_{u \in S^c} d(u)$; then, observing that $\sum_{v \in S} d(v) = \text{vol}(S)$ and $\sum_{u \in S^c} d(u) = \text{vol}(S^c)$ allows us to simplify the expression: $\mathbf{y}^T\mathbf{Dy} = \frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)}$.

At last, we can combine these two components to write

$$\mathbf{y}^T\mathbf{Ly}/\mathbf{y}^T\mathbf{Dy} = \left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)}\right)^2 \text{cut}(S) / \left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)}\right) \tag{28}$$

$$= \text{cut}(S) \left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)}\right). \tag{29}$$

Finally, note that for any two numbers A, B we have $A + B \leq 2\max\{A, B\}$, and so $\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)} \leq 2\max\left\{\frac{1}{\text{vol}(S)}, \frac{1}{\text{vol}(S^c)}\right\}$, which is the same as $1/\min\{\text{vol}(S), \text{vol}(S^c)\}$. Hence, $\mathbf{y}^T\mathbf{Ly}/\mathbf{y}^T\mathbf{Dy} \leq 2\,\text{cut}(S)/\min\{\text{vol}(S), \text{vol}(S^c)\}$, which exactly equals $2\phi(S)$. We have shown that this is true for *any* set S, and so we have shown in particular that $\mathbf{y}^T\mathbf{Ly}/\mathbf{y}^T\mathbf{Dy} \leq 2\Phi$, the conductance of the minimum conductance cut. But we showed above that that $R(\hat{\mathbf{L}}, \mathbf{x}) = \mathbf{y}^T\mathbf{Ly}/\mathbf{y}^T\mathbf{Dy}$, and so we've proved half of the theorem, $\lambda_{n-1} \leq 2\Phi$.

□

Techniques in the proof are useful for expressing and relating graph and matrix properties, and will come back again and again. Often constructing a contrived vector like $\mathbf{y}$ turns out to be a very useful trick. This relationship between $R(\hat{\mathbf{L}}, \mathbf{x})$ and conductance is essential, and gets used again and again.

5.1 Cauchy Interlacing Theorem

Min-max eigenvalue result can be used to prove that following theorem that helps us relate the eigenvalues of a graph to eigenvalues of its subgraphs. The Cauchy Interlacing Theorem can be stated in more generality so that it applies in slightly broader contexts linear algebraically speaking, but we don't need that extra strength for our purposes. Instead, we will state a version of Cauchy Interlacing Theorem [3] that is particular to our purposes. Before stating the theorem, we first define a *principal submatrix* $\mathbf{B}$ of a $n \times n$ matrix $\mathbf{A}$ as follows: let $j_1, ...j_m$ be any indices in $[1, n]$. Then a principal submatrix is any submatrix of the form $\mathbf{B} = \mathbf{A}([j_1, \cdots, j_n], [j_1, \cdots, j_n])$. In words, $\mathbf{B}$ is a subset of the rows (and columns corresponding to those rows) of $\mathbf{A}$.

Theorem 6 (Cauchy's Interlacing Theorem)**.** *Let $\boldsymbol{A}$ be a real-valued, symmetric $n \times n$ matrix. Let $\boldsymbol{B}$ be any $m \times m$ principal submatrix of $\boldsymbol{A}$, with $n = m + k$; i.e., let $\boldsymbol{B}$ result from removing k rows and their corresponding columns from $\boldsymbol{A}$. Let $\boldsymbol{A}$ have eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and $\boldsymbol{B}$ have eigenvalues $\mu_1 \geq \cdots \geq \mu_m$. Then these eigenvalues satisfy*

$$\lambda_{k+j} \leq \mu_j \leq \lambda_j$$

for $j = 1 : m$.

5.1.1 Exercises

5.1 Proving Lemma 3. Note that $\mathbf{f}^T\mathbf{Lf} = \mathbf{f}^T\mathbf{Df} - \mathbf{f}^T\mathbf{Af}$.

(a) Show that the sum $\mathbf{f}^T\mathbf{Af} = \sum_{i=1}^n \sum_{j=1}^n f_i A_{ij} f_j$ is equal to $2\sum_{e_{ij} \in E} f_i f_j$.

(b) Fix any node k. Given that node k has $d(k)$ edges incident to it, show that in the summation $\sum_{e_{ij} \in E} f_i f_j$ the factor f_k appears in $d(k)$ different terms of the summation.

(c) Next show that in the sum $\mathbf{f}^T\mathbf{Df}$ the term f_k^2 appears a total of $d(k)$ times.

(d) Use the first two parts, and the fact that $f_i^2 + f_j^2 - 2f_i f_j = (f_i - f_j)^2$, to show that $\mathbf{f}^T\mathbf{Lf} = \sum_{e_{ij} \in E} (f_i - f_j)^2$.

References

[1] Vladimir Batagelj and Matjaz Zaversnik, *An o (m) algorithm for cores decomposition of networks*, arXiv preprint cs/0310049 (2003).

[2] Fan RK Chung, *Spectral graph theory*, vol. 92, American Mathematical Soc., 1997.

[3] Gene H Golub and Charles F Van Loan, *Matrix computations*, vol. 3, JHU Press, 2012.

[4] Ryan A Rossi, David F Gleich, Assefaw H Gebremedhin, and Md Mostofa Ali Patwary, *Fast maximum clique algorithms for large graphs*, Proceedings of the 23rd International Conference on World Wide Web, ACM, 2014, pp. 365–366.