Lecture 1: Introduction

In which we describe what this course is about, we introduce the Laplacian matrix of a graph, and we prove our first result of spectral graph theory.

1 Overview

This is a class on graph theory and on graph algorithms, with an emphasis on the use of methods from linear algebra and from convex optimization. In the finite-dimensional case, linear algebra deals with vectors and matrices, and with a number of useful concepts and algorithms, such as determinants, eigenvalues, eigenvectors, and solutions to systems of linear equations.

The application to graph theory and graph algorithms comes from associating, in a natural way, a matrix to a graph $G = (V, E)$, and then interpreting the above concepts and algorithms in graph-theoretic language. The most natural representation of a graph as a matrix is via the $|V| \times |V|$ *adjacency matrix* of a graph, and certain related matrices, such as the *Laplacian* and *normalized Laplacian* matrix will be our main focus. We can think of $|V|$-dimensional Boolean vectors as a representing a partition of the vertices, that is, a *cut* in the graph, and we can think of arbitrary vectors as *fractional* cuts. From this point of view, eigenvalues are the optima of continuous relaxations of certain cut problems, the corresponding eigenvectors are optimal solutions, and connections between spectrum and cut structures are given by rounding algorithms converting fractional solutions into integral ones. When we take this view, we can naturally ask what other convex optimization problems, besides the calculation of eigenvalues and eigenvectors, can be used to reason about cuts and clusters in graphs, and this will lead us to the theory of semidefinite programming, a convex optimization technique that generalizes both linear programming and eigenvalue computations.

When we study random graphs, the matrices that we associate to them, such as the adjacency matrix and the Laplacian matrix, will be random matrices. We will cover some basic results from random matrix theory, which are very useful general results to keep in your mathematical toolbox, and we will see how to apply such results to the average-case analysis of spectral algorithms.

An important property of random graphs is that they are *expander graphs*. We will study the connection between expansion, random walks and other useful properties, and we will see how to calculate the spectrum of Cayley graphs and how to combine Cayley graphs to construct expanders. Time permitting, we will look at percolation in graphs, a random process whose study ties together many of the techniques that we introduce in this course.

The course can be roughly subdivided into four parts: in the first two parts of the course we will study *spectral graph algorithms*, that is, graph algorithms that make use of eigenvalues and eigenvectors of the normalized Laplacian of the given graph, and we will derive both worst-case analyses (in the first part) and average-case analyses (in the second part) of such algorithms. In the third part of the course we will study semidefinite programming, and we will look at it as a way of making spectral techniques more robust, in a sense that we will make precise. In the last part of the course we will look at graph expansion and graph percolation.

1.1 Spectral Graph Algorithms

We will study approximation algorithms for the *sparsest cut* problem, in which one wants to find a cut (a partition into two sets) of the vertex set of a given graph so that a minimal number of edges cross the cut compared to the number of pairs of vertices that are disconnected by the removal of such edges.

This problem is related to estimating the edge expansion of a graph and to find *balanced separators*, that is, ways to disconnect a constant fraction of the pairs of vertices in a graph after removing a minimal number of edges.

Finding balanced separators and sparse cuts arises in *clustering* problems, in which the presence of an edge denotes a relation of similarity, and one wants to partition vertices into few clusters so that, for the most part, vertices in the same cluster are similar and vertices in different clusters are not. For example, sparse cut approximation algorithms are used for *image segmentation*, by reducing the image segmentation problem to a graph clustering problem in which the vertices are the pixels of the image and the (weights of the) edges represent similarities between nearby pixels.

Balanced separators are also useful in the design of divide-and-conquer algorithms for graph problems, in which one finds a small set of edges that disconnects the graph, recursively solves the problem on the connected components, and then patches the partial solutions and the edges of the cut, via either exact methods (usually dynamic programming) or approximate heuristic. The sparsity of the cut determines the running time of the exact algorithms and the quality of approximation of the heuristic ones.

We will study a spectral algorithm first proposed by Fiedler in the 1970s, and to put its analysis into a broader context, we will also study the Leighton-Rao algorithm,

which is based on linear programming, and the Arora-Rao-Vazirani algorithm, which is based on semidefinite programming. We will see how the three algorithms are based on conceptually similar continuous relaxations.

Before giving the definition of sparsest cut, it is helpful to consider examples of graphs that have very sparse cuts, in order to gain intuition.

Suppose that a communication network is shaped as a path, with the vertices representing the communicating devices and the edges representing the available links. The clearly undesirable feature of such a configuration is that the failure of a single edge can cause the network to be disconnected, and, in particular, the failure of the middle edge will disconnect half of the vertices from the other half.

This is a situation that can occur in reality. Most of Italian highway traffic is along the highway that connect Milan to Naples via Bologna, Florence and Rome. The section between Bologna and Florence goes through relatively high mountain passes, and snow and ice can cause road closures. When this happens, it is almost impossible to drive between Northern and Southern Italy. This is not an Italian exception: I was once driving from Banff, a mountain resort town in Alberta which hosts a mathematical institute, back to the US. Suddenly, traffic on Canada's highway 1 came to a stop. People from the other cars, after a while, got out of the cars and started hanging out and chatting on the side of the road. We asked if there was any other way to go in case whatever accident was ahead of us would cause a long road closure. They said no, this is the only highway here. Thankfully we started moving again in half an hour or so.

Now, consider a two-dimensional $\sqrt{n} \times \sqrt{n}$ grid. The removal of an edge cannot disconnect the graph, and the removal of a constant number of edges can only disconnected a constant number of vertices from the rest of the graph, but it is possible to remove just $\sqrt{n}$ edges, a $1/O(\sqrt{n})$ fraction of the total, and have half of the vertices be disconnected from the other half.

A k-dimensional hypercube with $n = 2^k$ is considerably better connected than a grid, although it is still possible to remove a vanishingly small fraction of edges (the edges of a dimension cut, which are a $1/k = 1/\log_2 n$ fraction of the total number of edges) and disconnect half of the vertices from the other half.

Clearly, the most reliable network layout is the clique; in a clique, if an adversary wants to disconnect a p fraction of vertices from the rest of the graph, he has to remove at least a $p \cdot (1 - p)$ fraction of edges from the graph.

This property of the clique will be our "gold standard" for reliability. The expansion and the sparsest cut parameters of a graph measure how worse a graph is compared with a clique from this point of view.

For simplicity, here we will give definitions that apply only to the case of regular graphs.

Definition 1 (Edge expansion of a set) *Let $G = (V, E)$ be a d-regular graph, and $S \subseteq V$ a subset of vertices. The edge expansion of S is*

$$\phi(S) := \frac{E(S, V - S)}{d|S|}$$

where $E(S, V - S)$ is the number of edges in E that have one endpoint in S and one endpoint in $V - S$.

$d|S|$ is a trivial upper bound to the number of edges that can leave S, and so $\phi(S)$ measures how much smaller the actual number of edges is than this upper bound. We can also think of $\phi(S)$ as the probability that, if we pick a random node v in S and then a random neighbor w of v, the node w happens to be outside of S.

The quantity $1 - \phi(S)$ is the average fraction of neighbors that vertices in S have within S. For example, if G represents a social network, and S is a subset of users of expansion .3, this means that, on average, the users in S have 70% of their friends within S.

If $(S, V - S)$ is a cut of the graph, and $|S| \leq |V - S|$, then $\phi(S)$ is, within a factor of two, the ratio between the fraction $E(S, V - S)/|E| = 2E(S, V - S)/dn$ of edges that we have to remove to disconnect S from $V - S$, and the fraction $|S| \cdot |V - S|/\binom{n}{2}$ of pairs of vertices that become unreachable if we do so. We define the edge expansion of a cut as

$$\phi(S, V - S) := \max\{\phi(S), \phi(V - S)\}$$

The edge expansion of a graph is the minimum of the edge expansion of all cuts.

Definition 2 (Edge expansion of a graph) *Let $G = (V, E)$ be a d-regular graph, its edge expansion is*

$$\phi(G) := \min_{S:0<|S|<|V|} \phi(S, V - S) = \min_{S:0<|S|\leq \frac{|V|}{2}} \phi(S)$$

If A is the adjacency matrix of a d-regular graph $G = (V, E)$, then the *normalized Laplacian* of G is the matrix $L := I - \frac{1}{d}A$. We will prove the Cheeger inequalities: that if $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of L, counted with multiplicities and sorted in nondecreasing order, then

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}$$

The lower bound $\phi(G) \geq \frac{\lambda_2}{2}$ follows by using the variational characterization of eigenvalues to think of λ_2 as the optimum of a continuous optimization problem, and then realizing that, from this point of view, λ_2 is actually the optimum of a *relaxation* of $\phi(G)$.

The upper bound $\phi(G) \leq \sqrt{2\lambda_2}$ has a constructive proof, showing that the set S_F returned by Fiedler's algorithm has size $\leq |V|/2$ and satisfies $\phi(S_F) \leq 2\sqrt{\lambda_2}$. The two inequalities, combined, show that $\phi(S_F) \leq 2\sqrt{\phi(G)}$ and provide a (tight) worst-case analysis of the quality of the cut found by Fiedler-s algorithm, compared with the optimal cut.

To put this result in a broader context, we will see the Leighton-Rao approximation algorithm, based on linear programming, which finds a cut of expansion $\leq \phi(G) \cdot O(\log |V|)$, and the Arora-Rao-Vazirani algorithm, based on semidefinite programming, which finds a cut of expansion $\leq \phi(G) \cdot O(\sqrt{\log |V|})$. The spectral, linear programming, and semidefinite programming relaxation can all be seen as very related.

We will then consider combinatorial characterizations, and algorithms for other laplacian eigenvalues.

We will prove a "higher order" Cheeger inequality that characterizes λ_k for $k \geq 2$ similarly to how the standard Cheeger inequality characterizes λ_2, and the proof will provide a worst-case analysis of spectral partitioning algorithms similarly to how the proof of the standard Cheeger inequality provides a worst-case analysis of Fiedler's algorithm.

The outcome of these results is that small Laplacian eigenvalues characterize the presence of sparse cuts in the graph. Analogously, we will show that the value of λ_n characterizes large cuts, and the proof a Cheeger-type inequality for λ_n will lead to the worst-case analysis of a spectral algorithm for max cut.

1.2 Finding Sparse Cuts and Cliques in Random Graphs

Complementing the *worst-case analysis* of spectral algorithms that we will see in the first part of the course, we will then see how to perform an *average-case analysis* of graph algorithms, assuming that the graph comes from certain natural and interesting probabilistic generative models. We will focus on two models: the *Stochastic Block Model* and the *Planted Clique* model.

In the special case that we will focus on, the Stochastic Block Model generates an n-vertex graph with a balanced sparse cut in the following way: we first randomly partition the vertices into two subsets of equal size, and then we sample random edges so that each pair of vertices on the same side of the partition is joined by an edge with probability p, and each pair of vertices on different sides of the partition is joined by an edge with probability q, where $p > q$. This models a network in which there are two types of nodes, and nodes of the same type are more likely to be joined by an edge. An important problem is, given a graph sampled from the Stochastic Block Model, to recover the partition, and hence correctly recover the "type" of each node.

For a fixed partition, the expectation of the adjacency matrix is (up to a small adjustment) a rank-2 matrix where the largest eigenvalue is $(p+q) \cdot n/2$, the second largest is $(p-q) \cdot n/2$ and all the other eigenvalues are zero; furthermore, the eigenvector of the second eigenvalue is (up to a shift) a Boolean vector that indicates the type of each vertex. The key application of spectral algorithms here is that the adjacency matrix of the graph is a random matrix, and we know very strong concentration results for random matrices which tell us that a random matrix is usually very close, in operator norm, to its expectation. This will allow us to infer that the eigenvector of the second eigenvalue of the adjacency matrix that we see is going to be close to the eigenvector of the second eigenvalue of the average adjacency matrix, which, as we just said, gives us the partition and the types of the vertices.

In the Planted Clique distribution, we pick a random Erdös-Renyi graph and then choose a random subset of nodes and turn it into a clique. The goal is to find this "planted" clique. This is a special case of a number of inference problems and unsupervised machine learning problems. We can again reason about the expected adjacency matrix for a fixed choice of clique vertices, and see that its spectrum gives away the clique, and then reason about concentration of the empirical adjacency matrix around its expectation.

1.3 Semidefinite Programming

Semidefinite Programming is a class of convex optimization problems that generalizes both linear programming and the computation of eigenvalues of Hermitian matrices (and, in particular, the spectral methods discussed above). There are several applications of Semidefinite Programming to graph theory and graph algorithms, and we will limit ourselves to the setting of the average-case analysis of algorithms for the Stochastic Block Model and the Planted Clique problems.

In a nutshell, the improvements coming from Semidefinite Programming in these particular examples will come from the fact that Semidefinite Programming gives good approximations of the ℓ_∞-to-ℓ_1 operator norm of a matrix and from the fact that random matrices concentrate around their expectation in a stronger way under this norm than under the ℓ_2-to-ℓ_2 operator norm which is the one controlling the performance of spectral algorithms. Furthermore, the ℓ_∞-to-ℓ_1 norm is more robust to outliers than the ℓ_2-to-ℓ_2 norm, and this will translate into more robust algorithms.

1.4 Expansion, Random Walks, and Percolation

1.4.1 Constructions and Applications of Expander Graphs

A family of constant-degree expanders is a collection of arbitrarily large graphs, all of degree $O(1)$ and edge expansion $\Omega(1)$. Expanders are useful in several applications, and a common theme in such applications is that even though they are sparse, they have some of the "connectivity" properties of a complete graph.

For example, if one removes a $o(1)$ fraction of edges from an expander, one is left with a connected component that contains a $1 - o(1)$ fraction of vertices.

Lemma 3 *Let $G = (V, E)$ be a regular graph of expansion ϕ. Then, after an $\epsilon < \phi$ fraction of the edges are adversarially removed, the graph has a connected component that spans at least $1 - \epsilon/2\phi$ fraction of the vertices.*

PROOF: Let d be the degree of G, and let $E' \subseteq E$ be an arbitrary subset of $\leq \epsilon|E| = \epsilon \cdot d \cdot |V|/2$ edges. Let $C_1, \ldots, C_m$ be the connected components of the graph $(V, E - E')$, ordered so that $|C_1| \geq |C_2| \geq \cdots \geq |C_m|$. We want to prove that $|C_1| \geq |V| \cdot (1 - 2\epsilon/\phi)$. We have

$$|E'| \geq \frac{1}{2} \sum_{i \neq j} E(C_i, C_j) = \frac{1}{2} \sum_i E(C_i, V - C_i)$$

If $|C_1| \leq |V|/2$, then we have

$$|E'| \geq \frac{1}{2} \sum_i d \cdot \phi \cdot |C_i| = \frac{1}{2} \cdot d \cdot \phi \cdot |V|$$

but this is impossible if $\phi > \epsilon$.

If $|C_1| \geq |V|/2$, then define $S := C_2 \cup \cdots \cup C_m$. We have

$$|E'| \geq E(C_1, S) \geq d \cdot \phi \cdot |S|$$

which implies that $|S| \leq \frac{\epsilon}{2\phi} \cdot |V|$ and so $C_1 \geq \left(1 - \frac{\epsilon}{2\phi}\right) \cdot |V|$. □

In a d-regular expander, the removal of k edges can cause at most $O(k/d)$ vertices to be disconnected from the remaining "giant component." Clearly, it is always possible to disconnect k/d vertices after removing k edges, so the reliability of an expander is essentially best possible.

Another way in which expander graphs act similarly to a complete graph is the following. Suppose that, given a graph $G = (V, E)$, we generate a sequence $v_1, \ldots, v_k$

by choosing $v_1 \in V$ uniformly at random and then performing a $(k-1)$-step random walk. If G is a complete graph (in which every vertex has a self-loop), this process uses $k \log |V|$ random bits and generates k uniform and independent random vertices. In an expander of constant degree, the process uses only $\log |V| + O(k)$ random bits, and the resulting sequence has several of the useful statistical properties of a sequence generated uniformly at random. Especially in the case in which k is of the order of $\log |V|$, using $O(\log |V|)$ instead of $O(\log^2 |V|)$ random bits can be a significant saving in certain application. (Note, in particular, that the sample space has polynomial size instead of quasi-polynomial size.)

Constructions of constant-degree expanders are useful in a variety of applications, from the design of data structures, to the derandomization of algorithms, from efficient cryptographic constructions to being building blocks of more complex quasirandom objects.

There are two families of approaches to the explicit (efficient) construction of bounded-degree expanders. One is via algebraic constructions, typically ones in which the expander is constructed as a Cayley graph of a finite group. Usually these constructions are easy to describe but rather difficult to analyze. The study of such expanders, and of the related group properties, has become a very active research program. There are also combinatorial constructions, which are somewhat more complicated to describe but considerably simpler to analyze.

1.4.2 Mixing time of random walks

If one takes a random walk in a regular graph that is connected and not bipartite, then, regardless of the starting vertex, the distribution of the t-th step of the walk is close to the uniform distribution over the vertices, provided that t is large enough. It is always sufficient for t to be quadratic in the number of vertices; in some graphs, however, the distribution is near-uniform even when t is just poly-logarithmic, and, indeed, the time is at most $O\left(\frac{1}{\lambda_2} \log |V|\right)$, and thus it is at most logarithmic in expander graphs.

Among other applications, the study of the "mixing time" (the time that it takes to reach the uniform distribution) of random walks has applications to analyzing the convergence time of certain randomized algorithms.

The design of approximation algorithms for *combinatorial counting* problems, in which one wants to count the number of solutions to a given NP-type problem, can be reduced to the design of *approximately uniform sampling* in which one wants to approximately sample from the set of such solutions. For example, the task of approximately counting the number of perfect matchings can be reduced to the task of sampling almost uniformly from the set of perfect matchings of a given graph. One can design approximate sampling algorithms by starting from an arbitrary solution and then

making a series of random local changes. The behavior of the algorithm then corresponds to performing a random walk in the graph that has a vertex for every possible solution and an edge for each local change that the algorithm can choose to make. Although the graph can have an exponential number of vertices in the size of the problem that we want to solve, it is possible for the approximate sampling algorithm to run in polynomial time, provided that a random walk in the graph converges to uniform in time poly-logarithmic in its size.

The study of the mixing time of random walks in graphs is thus a main analysis tool to bound the running time of approximate sampling algorithms (and, via reductions, of approximate counting algorithms).

As a way of showing applications of results proved so far, we will show that, because of Cheeger's inequality, the mixing time is upper-bounded by $O\left(\frac{1}{\phi^2}\log|V|\right)$, and then we will use the dual of the Leighton-Rao relaxation to show that $1/\phi$ can be upper-bounded by the congestion of a certain flow problem. We will apply this theory to the analysis of an algorithm that approximates the number of perfect matchings in a given dense bipartite graph.

1.4.3 Percolation

Given a graph $G = (V, E)$ and a parameter $0 < p < 1$, the bond percolation process applied to G is the process in which each edge of G is kept with probability p and deleted with probability $1-p$; the choices for different edges are mutually independent. The main question about the percolation process is whether the residual graph has a large connected component. This problem originates in mathematical physics and it has found application in the study of network reliability, in the study of influence in social networks, and in epidemiological models.

We will see how to use spectral techniques, expansion, and properties of random walks to recover some of the basic results from the theory of percolation.

2 The Basics of Spectral Graph Theory

Given an undirected graph $G = (V, E)$, the approach of spectral graph theory is to associate a symmetric real-valued matrix to G, and to related the eigenvalues of the matrix to combinatorial properties of G.

For the sake of this lecture, we will restrict ourselves to the case in which G is a d-regular graph, and we will then see how to extend our results to apply to irregular graphs as well.

The most natural matrix to associate to G is the adjacency matrix A such that

$A_{i,j} = 1$ if $\{i,j\} \in E$ and $A_{i,j} = 0$ otherwise. In the second part of the course, in which we will study expander graphs, the adjacency matrix will indeed be the most convenient matrix to work with. For the sake of the algorithms that we will analyze in the first part of the course, however, a slight variation called the *normalized Laplacian* is more convenient.

There are a few ways to motivate the definition of the Laplacian. One way is the following: the variational characterization of the eigenvalues of real symmetric matrices tells us that we can think of the eigenvalues of M as optima of min-max optimization problems in which vectors $\mathbf{x} \in \mathbb{R}^V$ are feasible solutions and the cost function is the Rayleigh quotient

$$R_M(x) = \frac{\mathbf{x}^T M \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

We know that every homogeneous polynomial of degree 2 can be realized as $x^T M x$ for some matrix M, and,if we want to study cuts in a graph $G = (V, E)$, it makes sense to choose a matrix M such that

$$\mathbf{x}^T M \mathbf{x} = \sum_{\{u,v\} \in E} (x_u - x_v)^2$$

because, if $\mathbf{x} \in \{0,1\}^V$ is a Boolean vector, representing a cut in the graph, then the right-hand-side expression above is counting the number of edges that cross the cut, and so optimization problems with the above cost functions will be relaxations of cut problems.

Some calculations show that the matrix having such a property is $dI - A$, which is called the Laplacian matrix of G. Indeed, we can verify that

$$\mathbf{x}^T (dI - A) \mathbf{x} = \sum_{\{u,v\} \in E} (x_u - x_v)^2$$

because both expressions are easily seen to be equal to

$$\sum_v d x_v^2 - 2 \sum_{\{u,v\} \in E} x_u x_v$$

As we will see in a moment, the eigenvalues of $dI - A$ are in the range $[0, 2d]$, and it is not hard to see that their sum is dn, so it is convenient to divide the Laplacian matrix by d so that the range and the average values of the eigenvalues of the resulting matrix are independent of the degree. (This degree independence will make it possible to generalize results to the irregular case.)

We have thus reached the following definition.

Definition 4 (Normalized Laplacian) *The normalized Laplacian matrix of an undirected d-regular graph $G = (V, E)$ is $L := I - \frac{1}{d} A$.*

We shall now prove the following relations between the eigenvalues of L and certain purely combinatorial properties of G.

Theorem 5 *Let G be a d-regular undirected graph, let A be the adjacency matrix of G, and $L = I - \frac{1}{d} \cdot A$ be the normalized Laplacian matrix of G. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the real eigenvalues of L with multiplicities, in nondecreasing order. Then*

1. *$\lambda_1 = 0$ and $\lambda_n \leq 2$.*
2. *$\lambda_k = 0$ if and only if G has at least k connected components.*
3. *$\lambda_n = 2$ if and only if at least one of the connected components of G is bipartite.*

Note that the first two properties imply that the multiplicity of 0 as an eigenvalue is precisely the number of connected components of G.

PROOF: By the characterization of the Rayleigh quotient of L that we established above, and from the variational characterization of eigenvalues, we have

$$\lambda_1 = \min_{\mathbf{x} \in \mathbb{R}^n - \{\mathbf{0}\}} \frac{\sum_{\{u,v\} \in E} (x_u - x_v)^2}{d \sum_v x_v^2}$$

and so $\lambda_1 \geq 0$ because the Rayleigh quotient, being a ratio of sums of squares, is always non-negative.

If we take $\mathbf{1} = (1, \ldots, 1)$ to be the all-one vector, we see that its Rayleigh quotient is 0, and so 0 is the smallest eigenvalue of L, with $\mathbf{1}$ being one of the vectors in the eigenspace of 1.

We also have the following formula for λ_k:

$$\lambda_k = \min_{S \ k-\text{dimensional subspace of } \mathbb{R}^n} \ \max_{\mathbf{x} \in S - \{\mathbf{0}\}} \frac{\sum_{\{u,v\} \in E} (x_u - x_v)^2}{d \sum_v x_v^2}$$

So, if $\lambda_k = 0$, there must exist a k-dimensional space S such that for every $\mathbf{x} \in S$ and every $\{u, v\} \in E$, we have $x_u = x_v$, and so $x_u = x_v$ for every u, v which are in the same connected component. This means that each $\mathbf{x} \in S$ must be constant within each connected component of G, and so the dimension of S can be at most the number of connected components of G, meaning that G has at least k connected components.

Conversely, if G has at least k connected components, we can let S be the space of vectors that are constant within each component, and S is a space of dimension at least k such that for every element $\mathbf{x}$ of S we have

$$\sum_{\{u,v\} \in E} (x_u - x_v)^2 = 0$$

meaning that S is a witness of the fact that $\lambda_k = 0$.

Finally, to study λ_n, we first note that we have the formula

$$\lambda_n = \max_{\mathbf{x} \in \mathbb{R}^n - \{\mathbf{0}\}} \frac{\mathbf{x}^T L \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

from the variational characterization of eigenvalues (see Handout 0).

We also observe that for every vector $\mathbf{x} \in \mathbb{R}^n$ we have

$$2\mathbf{x}^T \mathbf{x} - \mathbf{x}^T L \mathbf{x} = \frac{1}{d} \sum_{\{u,v\} \in E} (x_u + x_v)^2$$

and so

$$\lambda_n = 2 - \min_{\mathbf{x} \in \mathbb{R}^n - \{\mathbf{0}\}} \frac{\sum_{\{u,v\} \in E} (x_u + x_v)^2}{d \sum_v x_v^2}$$

from which it follows that

$$\lambda_n \leq 2$$

and if $\lambda_n = 2$ then there must be a non-zero vector $\mathbf{x}$ such that

$$\sum_{\{u,v\} \in E} (x_u + x_v)^2 = 0$$

which means that $x_u = -x_v$ for every edge $(u, v) \in E$.

Let us now define $A := \{v : x_v > 0\}$ and $B := \{v : x_v < 0\}$. The set $A \cup B$ is non-empty (otherwise we would have $\mathbf{x} = \mathbf{0}$) and is either the entire graph, or else it is disconnected from the rest of the graph, because otherwise an edge with an endpoint in $A \cup B$ and an endpoint in $V - (A \cup B)$ would give a positive contribution to $\sum_{\{u,v\} \in E} (x_u - x_v)^2$; furthermore, every edge incident on a vertex on A must have the other endpoint in B, and vice versa. Thus, $A \cup B$ is a connected component, or a collection of connected components, of G which is bipartite, with the bipartition A, B. $\square$