Chapter 3

Spectral graph theory and random walks on graphs

Algebraic graph theory is a major area within graph theory. One of the main themes of algebraic graph theory comes from the following question: what do matrices and linear algebra tell us about graphs? One of the most useful invariants of a matrix to look in linear algebra at are its eigenvalues and eigenspaces. This is also true in graph theory, and this aspect of graph theory is known as *spectral graph theory*.

Given a graph G, the most obvious matrix to look at is its adjacency matrix A, however there are others. Two other common choices are the *Laplacian matrix*, motivated from differential geometry, and what we will call the *transition matrix*, motivated from dynamical systems—specifically random walks on graphs. The Laplacian and transition matrices are closely related (the eigenvalues of one determine the eigenvalues of the other), and which to look at just depends upon one's point of view.

We will first examine some aspects of the adjacency matrix, and then discuss random walks on graphs and transition matrices. On one hand, eigenvalues can be used to measure how good the network flow is and give bounds on the diameter of a graph. This will help us answer some of the questions we raised in the first chapter about communication and transportation networks. On the other hand, eigenvectors will tell us more precise information about the flow of a networks. One application is using eigenvectors to give a different kind of centrality ranking, and we will use this idea when we explain Google PageRank.

Here are some other references on these topics.

- **Algebraic Graph Theory**, by Norman Biggs.
- **Algebraic Graph Theory**, by Chris Godsil and Gordon Royle.
- **Modern Graph Theory**, by Béla Bollobás.
- **Spectra of Graphs**, by Andries Brouwer and Willem Haemers.
- **Spectral Graph Theory**, by Fan Chung.

The first two books are "classical graph theory" books in the sense that they do not discuss random walks on graphs, and cover more than just spectral theory. Bollobás's book covers many

topics, including some spectral theory and random walks on graphs (and random graphs). The latter two books focus on spectral theory. Brouwer–Haemers cover the adjacency and Laplacian spectra but does not really discuss random walks, whereas Chung's book discusses random walks but focuses entirely on the (normalized) Laplacian matrix.

3.1 A review of some linear algebra

Before we get started on applying some linear algebra to graph theory, we will review some standard facts from linear algebra. This is just meant to be a quick reminder of the relevant theory, as well as fixing some notation, but you should refer to a linear algebra text to make sense of what I'm saying.

Let $M_n(\mathbb{R})$ denote the set of real $n \times n$ matrices, and $M_n(\mathbb{C})$ denote the set of complex $n \times n$ matrices. In what follows, we mainly work with real matrices, though most of the theory is the same for complex matrices.

Let $A \in M_n(\mathbb{R})$. We may view A as a linear operator

$$\begin{aligned} A : \mathbb{R}^n &\to \mathbb{R}^n \\ v &\mapsto Av \end{aligned}$$

where we view $v \in \mathbb{R}^n$ as a $n \times 1$ matrix, i.e., a column vector. We may sometimes write a column vector $v = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ as $v = (x_1, \dots, x_n)$ out of typographical convenience.

Let A^t denote the transpose of A. Then $(A^t)^t = A$. For $A, B \in M_n(\mathbb{R})$, we have $(AB)^t = B^t A^t$. Recall AB does not usually equal BA, though it happens from time to time.

Denote by $I = I_n$ the $n \times n$ identity matrix. This means $AI = IA = A$ for any $A \in M_n(\mathbb{R})$. We use 0 for the number 0 as well as zero matrices.

Denote by $\mathrm{diag}(d_1, \dots, d_n)$ the diagonal matrix with $A = (a_{ij})$ with $a_{ii} = d_i$ and $a_{ij} = 0$ if $i \neq j$. Note $I = \mathrm{diag}(1, 1, \dots, 1)$.

Recall A is invertible if there is a matrix $B \in M_n(\mathbb{R})$ such that $AB = I$. If there is such a matrix B, it must be unique, and we say B is the inverse of A and write $B = A^{-1}$. It is true that $AA^{-1} = A^{-1}A = I$, i.e., that the inverse of A^{-1} is just A, i.e., $(A^{-1})^{-1} = A$. For $A, B \in M_n(\mathbb{R})$ both invertible, we have that AB is also invertible and $(AB)^{-1} = B^{-1}A^{-1}$.

There is a function $\det : M_n(\mathbb{R}) \to \mathbb{R}$ called the determinant. It satisfies $\det(AB) = \det(A)\det(B)$, and A is invertible if and only if $\det A \neq 0$. We can define it inductively on n as follows. For $n = 1$, put $\det[a] = a$. Now $A = (a_{ij}) \in M_n(\mathbb{R})$ with $n > 1$. Let A_{ij} be the ij cofactor matrix, i.e., the matrix obtained by deleting the i-th row and j-th column. Then

$$\det(A) = a_{11} \det A_{11} - a_{12} \det A_{12} + \cdots + (-1)^{n-1} a_{1n} \det A_{1n}. \tag{3.1}$$

Hence the determinant can be computed by using a cofactor expansion along the top row. It can be computed similarly with a cofactor expansion along any row, and since $\det(A) = \det(A^t)$, it can be also computed using a cofactor expansion along any column. (Just remember to be careful of signs—the first sign is -1 if you start along an even numbered row or column.) For $n = 2$, we have

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc.$$

Example 3.1.1. *Let* $A = \begin{pmatrix} 2 & -1 & 0 \\ 0 & 2 & -1 \\ -1 & 0 & 2 \end{pmatrix}$. *Then*

$$\begin{aligned} \det A &= 2 \det \begin{pmatrix} 2 & -1 \\ 0 & 2 \end{pmatrix} - (-1) \det \begin{pmatrix} 0 & -1 \\ -1 & 2 \end{pmatrix} + 0 \det \begin{pmatrix} 0 & 2 \\ -1 & 0 \end{pmatrix} \\ &= 2 \cdot 4 + 1 \cdot (-1) + 0 = 7. \end{aligned}$$

For larger n, determinants are unpleasant to compute by hand in general, but some special cases can be worked out. For instance, if $D = \mathrm{diag}(d_1, \dots, d_n)$ is diagonal, then $\det D = d_1 d_2 \cdots d_n$. In particular, $\det I = 1$, and $\det kI_n = k^n$ for a scalar $k \in \mathbb{R}$. Also note that for invertible A, since $\det(A)\det(A^{-1}) = \det(AA^{-1}) = \det I = 1$, we have $\det(A^{-1}) = (\det A)^{-1}$. If S is invertible, then $\det(SAS^{-1}) = \det S \det A \det S^{-1} = \det A$. Hence similar matrices have the same determinant. Thus it makes sense to define the determinant $\det T$ of a linear transformation (i.e., this determinant does not depend upon a choice of basis).

Given any linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$, and any basis $\mathcal{B} = \{v_1, \dots v_n\}$, we can associate a matrix $[T]_{\mathcal{B}}$ in $M_n(\mathbb{R})$ such that if

$$T(x_1 v_1 + \cdots x_n v_n) = y_1 v_1 + \cdots + y_n v_n$$

then

$$[T]_{\mathcal{B}} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

In particular, let $\mathcal{B}_0 = \{e_1, \dots, e_n\}$ denote the standard basis for $\mathbb{R}^n$, i.e., e_i is the column vector with a 1 in the i-th position and 0's elsewhere. Then, if $Te_i = u_i$, the i-th column of $[T]_{\mathcal{B}_0}$ is u_i, i.e.,

$$[T]_{\mathcal{B}_0} = [u_1 | u_2 | \cdots | u_n].$$

Given any bases $\mathcal{B} = \{v_1, \dots, v_n\}$ and $\mathcal{B}' = \{v_1', \dots, v_n'\}$, there is an invertible change of basis matrix S, determined by requiring $Sv_i = v_i'$ for all i, such that

$$[T]_{\mathcal{B}'} = S[T]_{\mathcal{B}} S^{-1}. \tag{3.2}$$

For any matrices $A, B \in M_n(\mathbb{R})$, we say they are similar if $A = SAS^{-1}$ for an invertible $S \in M_n(\mathbb{R})$. This means A and B can be viewed as representing the same linear transformation T, just with respect to different bases.

We say a nonzero vector $v \in \mathbb{R}^n$ is a (real) eigenvector for A with (real) eigenvalue $\lambda \in \mathbb{R}$ if $Av = \lambda v$. If λ is an eigenvalue for A, the set of $v \in \mathbb{R}^n$ (including the zero vector) for which $Av = \lambda_v$ is called the λ-eigenspace for A, and it is a linear subspace of $\mathbb{R}^n$. Even if $A \in M_n(\mathbb{R})$ it may not have any real eigenvalues/vectors but it may have complex ones, so we will sometimes consider those. The dimension of the λ-eigenspace is called the geometric multiplicity of λ.

Eigenvalues and eigenvectors are crucial to understanding the geometry of a linear transformation as A having real eigenvalue λ means A simply acts as scaling by λ on the λ-eigenspace. Any complex eigenvalues must occur in pairs $re^{\pm i\theta}$, and complex eigenvalues means that A acts by scaling by r together with rotation by θ on an appropriate 2-dimensional subspace.

Eigenvalues can be computed as follows. If $Av = \lambda v = \lambda I v$ for a nonzero v, this means $(\lambda I - A)$ is not invertible, so it has determinant 0. The characteristic polynomial of A is

$$p_A(\lambda) = \det(\lambda I - A). \tag{3.3}$$

This is a polynomial of degree n in λ, and its roots are the eigenvalues. By the fundamental theorem of algebra, over $\mathbb{C}$ it always factors into linear terms

$$p_A(\lambda) = (\lambda - \lambda_1)^{m_1}(\lambda - \lambda_2)^{m_2} \cdots (\lambda - \lambda_k)^{m_k}. \tag{3.4}$$

Here $\lambda_1, \dots, \lambda_k$ denote the distinct (possibly complex) eigenvalues. The number m_i is called the (algebraic) multiplicity of λ_i. It is always greater than or equal to the (complex) dimension of the (complex) λ_i-eigenspace (the geometric multiplicity).

Now let us write the eigenvalues of A as $\lambda_1, \dots, \lambda_n$ where some of these may repeat. The spectrum of A is defined to be the (multi)set $\mathrm{Spec}(A) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$, where we include each eigenvalue with (algebraic) multiplicity. (Multiset is like a set, except that elements are allowed to appear with some multiplicity.) If $A = (a_{ij})$ is diagonal, then $\mathrm{Spec}(A) = \{a_{11}, a_{22}, \dots, a_{nn}\}$. From (3.4), we see the spectrum determines the characteristic polynomial, and vice versa.

Once we find the eigenvalues λ_i, we can find the eigenspaces by solving the system of equations $(\lambda I - A)v = 0$.

Suppose B is similar to A, e.g., $B = SAS^{-1}$. Then

$$p_B(\lambda) = \det(\lambda I - SAS^{-1}) = \det(S(\lambda I - A)S^{-1}) = \det(S)\det(\lambda I - A)\det(S)^{-1} = p_A(\lambda), \tag{3.5}$$

so A and B have the same eigenvalues and characteristic polynomials. (Their eigenspaces will be different, but can be related by a change of basis matrix.)

One can shew that the spectrum a block diagonal matrix $\begin{pmatrix} A & \\ & B \end{pmatrix}$ is the $\mathrm{Spec}(A) \cup \mathrm{Spec}(B)$. Black matrix entries represent zeroes.

We say A is diagonalizable if A is similar to a diagonal matrix D. A matrix A is diagonalizable if and only if there is a basis of $\mathbb{C}^n$ consisting of eigenvectors for A. This is always possible if A has n distinct eigenvalues, but not always possible when you have repeated eigenvalues (cf. Exercise 3.1.8).

If there is a basis of eigenvectors $\{v_1, \dots, v_n\}$, the matrix $S = [v_1|\cdots|v_n]$ will be invertible, and doing the associated change of basis will give us a diagonal matrix $D = SAS^{-1}$. Specifically, we will get $D = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ where λ_i is the eigenvalue associated to v_i. Doing this change of basis to get a diagonal matrix is called diagonalizing A.

One benefit of diagonalization is that it allows us to exponentiate easily. Suppose $D = SAS^{-1}$ is diagonal. Thinking of A as a linear transformation, there are many instances where we want to apply A repeatedly. We determine the resulting transformation by looking at powers of A. Note

$$A^m = (S^{-1}DS)^m = (S^{-1}DS)(S^{-1}DS)\cdots(S^{-1}DS) = S^{-1}D^mS. \tag{3.6}$$

For any two diagonal matrices $X = \mathrm{diag}(x_1, \dots, x_n)$ and $Y = \mathrm{diag}(y_1, \dots, y_n)$, we have $XY = YX = \mathrm{diag}(x_1y_1, \dots, x_ny_n)$. In particular, if $D = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$, then $D^m = \mathrm{diag}(\lambda_1^m, \dots, \lambda_n^m)$. In other words, D^m is easy to compute, so we may compute A^m easily.

There is some information we can easily get about eigenvalues without determining them exactly. First, suppose A is diagonalizable, i.e., $D = SAS^{-1}$ is diagonal for some S. Then

$D = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ where the λ_i's are the eigenvalues of A (with multiplicity). Consequently, $\det A = \det D = \lambda_1 \lambda_2 \cdots \lambda_n$. In other words, the determinant of a matrix A is the product of the eigenvalues (with multiplicity). In fact, this is true for any A, not just diagonalizable matrices A. (Any A is similar to an upper triangular matrix, whose diagonal entries must be the eigenvalues, and now apply Exercise 3.1.1.)

The trace is another useful invariant to study eigenvalues. The trace of a matrix $\mathrm{tr} A$ is the sum of its diagonal entries. Note the trace map satisfies $\mathrm{tr}(A+B) = \mathrm{tr}\, A + \mathrm{tr}\, B$. Unlike the determinant, it does not satisfy $\mathrm{tr}(AB) = \mathrm{tr}\, A \mathrm{tr}\, B$ (just like $\det(A+B) \neq \det A + \det B$ in general). Nevertheless, similar matrices have identical trace. Therefore, we see if A is diagonalizable, then $\mathrm{tr}\, A$ is the sum of its eigenvalues. This statement is also true for any $A \in M_n(\mathbb{R})$.

Here is one of the main theorems on diagonalizability, often just called the spectral theorem.

Theorem 3.1.2. *Let $A \in M_n(\mathbb{R})$ be symmetric, i.e., $A^t = A$. Then A has only real eigenvalues and is diagonalizable over $\mathbb{R}$. In fact, A is diagonalizable by an orthogonal matrix S, i.e., there exists $S \in M_n(\mathbb{R})$ with $SS^t = I$ and SAS^{-1} is diagonal.*

Exercises

Exercise 3.1.1. *Let $A = (a_{ij}) \in M_n(\mathbb{R})$ be upper triangular, i.e., $a_{ij} = 0$ whenever $i > j$.*
(i) Prove $\det A = a_{11} a_{22} \cdots a_{nn}$.
(ii) Use this to show that $\mathrm{Spec}(A) = \{a_{11}, a_{22}, \dots, a_{nn}\}$.

Exercise 3.1.2. *Analyze the running time of the algorithm to compute the determinant by recursively using the cofactor expansion (3.1),*

Exercise 3.1.3. *Let $A \in M_n(\mathbb{R})$. Suppose v is an eigenvector with eigenvalue λ for A. Show, for $m = 1, 2, 3, \dots$, that v is an eigenvector with eigenvalue λ^m for A^m.*

Exercise 3.1.4. *Let $A \in M_2(\mathbb{R})$. Show $p_A(\lambda) = \lambda^2 - \mathrm{tr}(A)\lambda + \det(A)$ in two different ways: (i) use the definition $p_A(\lambda) = \det(\lambda I - A)$, and (ii) abstractly write $p_A(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)$.*

Exercise 3.1.5. *Let $A \in M_n(\mathbb{R})$ and write $p_A(\lambda) = \lambda^n + c_1 \lambda^{n-1} + c_2 \lambda^{n-2} + \cdots + c_{n-1}\lambda + c_n$. Show $c_1 = -\mathrm{tr} A$ and $c_n = (-1)^n \det A$.*

Exercise 3.1.6. *Show that A is diagonalizable if and only if the geometric multiplicity of each eigenvalue λ equals the algebraic multiplicity.*

Exercise 3.1.7. *Diagonalize the rotation-by-θ matrix $A = \begin{pmatrix} \cos i\theta & \sin i\theta \\ -\sin i\theta & \cos i\theta \end{pmatrix}$.*

Exercise 3.1.8. *Find a 2×2 matrix which is not diagonalizable. (Hint: you need to show it does not have a basis of eigenvectors, so it must have a repeated eigenvalue. Try looking at upper triangular matrices and use Exercise 3.1.1.)*

Exercise 3.1.9. *Diagonalize the matrix $A = \frac{1}{2}\begin{pmatrix} 0 & 2 \\ 1 & 1 \end{pmatrix}$ and use this to write down a simple expression for A^m. Can you make sense of $\lim_{m \to \infty} A^m$?*

3.2 Rudiments of spectral graph theory

▶ Here we assume graphs are undirected and simple unless stated otherwise.

Adjacency spectra

Let's start of by making some elementary observations on adjacency matrices. Let $G = (V, E)$ be a graph on the ordered set $V = \{1, 2, \dots, n\}$ and A be the adjacency matrix with repect to V. Write $A = (a_{ij})$ so $a_{ij} = 1$ if and only if $(i, j) \in E$.

Proposition 3.2.1. *The (i, j) entry of the matrix A^ℓ is the number of paths of length ℓ from i to j in G. This is still true if G is directed and/or non-simple.*

Proof. This is clearly true when $\ell = 0$ or 1. Now we prove this by induction on ℓ. Suppose it is true for $\ell = m$. We show it is true for $\ell = m + 1$. Write $B = (b_{ij}) = A^m$ and $C = (c_{ij}) = A^{m+1}$. Writing $C = AB$, we see

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

By our induction hypothesis, a_{ik} is the number of paths of length 1 from i to k (which is 0 or 1) and b_{kj} is the number of paths of length m from k to j. Hence c_{ij} is the sum over all neighbors k of vertex i of the number of paths of length ℓ from k to j. This must be the total number of paths of length $m + 1$ from i to j. □

Let σ be a permutation of V, i.e., $\sigma : V \to V$ is a bijective map. Let $\mathcal{B}_0 = \{e_1, \dots, e_n\}$ be the standard basis for $\mathbb{R}^n$. Then we can also view σ as a permutation of $\mathcal{B}_0$ given by $\sigma(e_i) = e_{\sigma(i)}$. Thus we can express σ as a matrix $S = [\sigma]_{\mathcal{B}_0}$. This is a 0–1 matrix (a matrix whose entries are all either 0 or 1) with exactly one 1 in each row and column. Such matrices are called permutation matrices. Since σ is bijective, the associated matrix S must be invertible, as S^{-1} effects the inverse permutation σ^{-1}.

For instance, suppose $V = \{1, 2, 3, 4\}$ and σ is the "cyclic shift" of V given by $\sigma(1) = 4$, $\sigma(2) = 1$, $\sigma(3) = 2$ and $\sigma(4) = 3$. Then the corresponding matrix is

$$S = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

One readily checks $Se_1 = e_4$, $Se_2 = e_1$, $Se_3 = e_2$ and $Se_4 = e_3$. Note that S is the adjacency matrix for the directed cycle graph on 4 vertices where the directed cycle goes in the clockwise direction, and S^{-1} is the adjacency matrix for the directed cycle graph where the cycle goes in the counterclockwise direction.

Now let B be the adjacency matrix for G with respect to the ordering $\sigma(1), \sigma(2), \dots, \sigma(n)$. Then $B = SAS^{-1}$. Consequently all possible adjacency matrices for G are similar. Hence the quantities $\det A, \mathrm{tr} A$ and $\mathrm{Spec}(A)$ and $p_A(\lambda)$ are all invariants of G, i.e., they do not depend upon the labelling of vertices. Thus the following definition makes sense.

Definition 3.2.2. *Let G be a (possibly directed and non-simple) graph, and A be any adjacency matrix for G. We say λ is an* **(adjacency) eigenvalue** *for G if λ is an eigenvalue for A, and the* **(adjacency) spectrum** *of G is $\mathrm{Spec}\, G = \mathrm{Spec}\, A$. The* **characteristic polynomial** *of G is $p_G(\lambda) = p_A(\lambda)$. We also define the* **determinant** *and* **trace** *of G by $\det G = \det A$ and $\mathrm{tr}\, G = \mathrm{tr}\, A$.*

However, the eigenspaces of A in general depend upon the labelling (or ordering of vertices), so eigenspaces/eigenvectors are not invariants of G. However, if $Av = \lambda v$, then

$$BSv = (SAS^{-1})(Sv) = SAv = S\lambda v = Sv. \tag{3.7}$$

That is, we can go from eigenvectors of A to eigenvectors of B by applying the permutation matrix S (which just permutes the entries of v). This will be important for us later.

While isomorphic graphs, i.e., graphs that differ only up to labelling of vertices, must have the same spectrum, knowing that $\operatorname{Spec} G_1 = \operatorname{Spec} G_2$ does not imply that G_1 and G_2 are isomorphic. Since one can compare spectra in polynomial time, if this were true it would give us a polynomial time algorithm to answer the graph isomorphism problem. However, it is conjectured that the spectrum determines the isomorphism class of the graph "most of the time." If true, this means for "most" G_1, there is a polynomial-time algorithm that can determine if any G_2 is isomorphic to G_1.* Much work is currently being done to determine which graphs are determined by their spectra.

Spectral graph theory (which includes the above conjecture) is the study of what information about graphs we can read off from the spectrum (and related data). For instance, the number of elements (with multiplicity) in $\operatorname{Spec}(G)$ determines the order n of G. Spectral graph theory is an extremely active area of research, and we will just present as sample of some basic results in this section to motivate its utility.

Proposition 3.2.3. *Spec(G) determines the number of closed paths of length ℓ for all ℓ, and this number is* $\operatorname{tr}(A^\ell)$. *Here G may be directed and non-simple.*

Proof. Note the number of closed paths of length ℓ is simply $\operatorname{tr}(A^\ell)$. If $\operatorname{Spec}(A) = \{\lambda_1, \dots, \lambda_n\}$, then $\operatorname{Spec}(A^\ell) = \{\lambda_1^\ell, \dots, \lambda_n^\ell\}$ by Exercise 3.1.3. Hence $\operatorname{tr}(A^\ell) = \sum_i \lambda_i^\ell$ is determined by $\operatorname{Spec}(G) = \operatorname{Spec}(A)$. □

In fact, one can rewrite the characteristic polynomial in terms of $\operatorname{tr}(A^\ell)$ for $1 \le \ell \le n$, and deduce the converse of this proposition, but we will not do this.

We can be more explicit about the traces $\operatorname{tr}(A^\ell)$ of the first few powers of A.

Proposition 3.2.4. *Write* $\operatorname{Spec}(G) = \{\lambda_1, \dots, \lambda_n\}$. *Then we have each $\lambda_i \in \mathbb{R}$ and*
(i) $\operatorname{tr}(A) = \lambda_1 + \lambda_2 + \cdots + \lambda_n = 0$;
(ii) $\frac{1}{2}\operatorname{tr}(A^2) = \frac{1}{2}(\lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2)$ *is the number of edges of G; and*
(iii) $\frac{1}{6}\operatorname{tr}(A^3) = \frac{1}{6}(\lambda_1^3 + \lambda_2^3 + \cdots + \lambda_n^3)$ *is the number of triangles in G.*

Proof. Each eigenvalue is real by the spectral theorem.

(i) The trace is zero because each diagonal entry of A is 0. For non-simple graphs, $\operatorname{tr}(A)$ will just be the total number of loops.

(ii) Each closed path of length 2 is simply means traversing an edge (u, v) going back, and each edge (u, v) corresponds to 2 closed paths of length 2.

(iii) is similar to (ii). Any closed path of length 3 must be a 3-cycle, and each triangle yields 6 different closed paths of length 3. □

*Of course, given any two random graphs G_1 and G_2, most of the time they will not be isomorphic and can be easily distinguished in polynomial time by simple invariants such as degree distributions. The strength of this statement is that if G_1 lies outside of a conjecturally small class of graphs, then we get a polynomial-time algorithm that works for *any* G_2.

These arguments no longer work for higher powers of traces because the closed paths of length ℓ will include more than just the of cycles of length ℓ for $\ell \geq 4$, but include paths with repeated vertices. However, see Exercise 3.2.2 for $\ell = 4$.

Since all eigenvalues of real, we will typically order them as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. By (ii), we see that all eigenvalues are 0 if and only if G has no edges (this is not true for directed graphs—see example below). From now on we will assume this is not the case. Then, as the sum of the eigenvalues is 0, we know $\lambda_1 > 0$ and $\lambda_n < 0$.

Now let us look at a few simple directed and undirected examples.

Example 3.2.5 (Directed path graphs)**.** *Let G be the directed path graph on n vertices. Then*

$$p_G(\lambda) = \det \begin{pmatrix} \lambda & -1 & 0 & \cdots & 0 \\ 0 & \lambda & -1 & \ddots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda & -1 \\ 0 & \cdots & \cdots & 0 & \lambda \end{pmatrix} = \lambda^n,$$

by Exercise 3.1.1. Hence all eigenvalues are 0.

Example 3.2.6 (Directed cycle graphs)**.** *Let G be the directed cycle graph on n vertices. Then*

$$p_G(\lambda) = \det \begin{pmatrix} \lambda & -1 & 0 & \cdots & 0 \\ 0 & \lambda & -1 & \ddots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda & -1 \\ -1 & 0 & \cdots & 0 & \lambda \end{pmatrix} = \lambda^n - 1.$$

This can be seen by using a cofactor expansion along the first column, and using Exercise 3.1.1 and its analogue for lower triangular matrices (true since $\det(A) = \det(A^t)$). Thus $\mathrm{Spec}(G) = \{1, \zeta, \zeta^2, \ldots, \zeta^{n-1}\}$ is the set of n-th roots of unity, where $\zeta = e^{2\pi i/n}$. Hence we may get complex eigenvalues (and also complex eigenvectors) for directed graphs.

Example 3.2.7 (Path graphs)**.** *Consider the path graph P_n on n vertices. Then*

$$p_{P_n}(\lambda) = \det \begin{pmatrix} \lambda & -1 & 0 & \cdots & 0 & 0 \\ -1 & \lambda & -1 & \ddots & 0 & 0 \\ 0 & -1 & -\lambda & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & -1 & \lambda & -1 \\ 0 & 0 & \cdots & & -1 & \lambda \end{pmatrix}$$

From a cofactor expansion, we see we have the recursion

$$p_{P_n}(\lambda) = \lambda p_{P_{n-1}}(\lambda) - p_{P_{n-2}}(\lambda), \quad n \geq 4.$$

Thus we can compute the first few cases

$$\begin{aligned} p_{P_2}(\lambda) &= \lambda^2 - 1 & \mathrm{Spec}(P_2) &= \{1, -1\} \\ p_{P_3}(\lambda) &= \lambda(\lambda^2 - 2) & \mathrm{Spec}(P_3) &= \{\sqrt{2}, 0, -\sqrt{2}\} \\ p_{P_4}(\lambda) &= \lambda^4 - 3\lambda^2 + 1 & \mathrm{Spec}(P_4) &= \{\sqrt{\frac{3+\sqrt{5}}{2}}, \sqrt{\frac{3-\sqrt{5}}{2}}, -\sqrt{\frac{3-\sqrt{5}}{2}}, -\sqrt{\frac{3+\sqrt{5}}{2}}\}. \end{aligned}$$

Example 3.2.8 (Cycle graphs)**.** *Consider a cycle graph $G = C_n$. Explicit calculations give*

$$\begin{aligned} p_{C_3}(\lambda) &= \lambda^3 - 3\lambda - 2 & \mathrm{Spec}(C_3) &= \{2, -1, -1\} \\ p_{C_4}(\lambda) &= \lambda^4 - 4\lambda^2 & \mathrm{Spec}(C_4) &= \{2, 0, 0, -2\}. \end{aligned}$$

It is easy to see in general that the all ones vector $(1, 1, \ldots, 1)$ is an eigenvector with eigenvalue 2. If n is even, one can check the vector with alternating entries $(1, -1, 1, -1, \ldots, 1, -1)$ is an eigenvector with eigenvalue 2. One can show that the other eigenvalues are of the form $2\cos(2j\pi/n)$ for $1 \leq j < \lfloor \frac{n}{2} \rfloor$, each with multiplicity 2. See Exercise 3.2.3 for the case of $n = 6$.

Example 3.2.9 (Complete graphs)**.** *Take $G = K_n$. Then the adjacency matrix (with respect to any ordering of vertices!) is*

$$A = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \cdots & 1 & 0 & 1 \\ 1 & \cdots & \cdots & 1 & 0 \end{pmatrix}.$$

In this case, due to the nature of A, the easiest way to find the eigenvalues is to guess a basis of eigenvectors. Clearly the all ones vector $v_1 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ is an eigenvector with eigenvalue $n - 1$. For $2 \leq i \leq n$, let v_i denote the vector with a 1 in position 1, -1 in position i, and 0's elsewhere. Then it is easy to see $Av_i = -v_i$ for $i \geq 2$. Hence -1 is an eigenvalue with multiplicity (geometric, and therefore also algebraic) $n - 1$. That is,

$$\mathrm{Spec}(G) = \{n - 1, -1, -1, \cdots, -1\},$$

and if you care

$$p_G(\lambda) = (\lambda - (n-1))(\lambda + 1)^{n-1}.$$

More generally, for a k-regular graph, the maximum eigenvalue will be k. We can also given upper bound for arbitrary (simple, undirected) graphs in terms of degree.

Proposition 3.2.10. *Let k be the maximum degree of a vertex in G. Then for any eigenvalue λ of G, $|\lambda| \leq k$. In particular, $|\lambda| \leq n - 1$. Moreover, if G is k-regular, then in fact $\lambda = k$ is an eigenvalue.*

Put another way, the maximum eigenvalue gives a lower bound for the maximum degree of G. And for regular graphs, the maximum eigenvalue determines the degree.

Proof. Let $v = (x_1, x_2, \ldots, x_n)$ be an eigenvector for G with eigenvalue λ. Say $|x_m|$ achieves the maximum of all $|x_j|$'s. By scaling, we may assume $x_m = 1$. Write $Av = \lambda v = (y_1, y_2, \ldots, y_n)$ and A_{ij} Then

$$|\lambda| = |\lambda x_m| = |y_m| = \left| \sum_{j=1}^{n} a_{mj} x_j \right| \leq k$$

since at most k of the entries a_{mj} are 1.

If G is k-regular, then all row sums of A are k, so $Av = kv$ where $v = (1, 1, \ldots, 1)$. □

In fact, for k-regular graphs G, the multiplicity of $\lambda = k$ equals number of connected components of G. More generally, to determine the spectrum of G it suffices to look at the connected components.

Lemma 3.2.11. *Let $G_1, \ldots, G_r$ denote the connected components of a graph G. Then*

$$\mathrm{Spec}(G) = \mathrm{Spec}(G_1) \cup \mathrm{Spec}(G_2) \cup \cdots \mathrm{Spec}(G_r).$$

Proof. We may order the vertices of G in such a way, that the adjacency matrix A for G is a block diagonal matrix of the form

$$A = \begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_r \end{pmatrix}$$

where A_i is an adjacency matrix for G_i. The eigenvalues of a block diagonal matrix are just the collection of the eigenvalues of the blocks, with multiplicity. □

This implies that if G is k-regular, and G has r connected components, the multiplicity of $\lambda = k$ is at least r. To conclude it is exactly r, we need to show that this multiplicity is 1 if G is connected. In fact, a similar argument will tell us something about the minimum eigenvalue too.

Proposition 3.2.12. *If G is k-regular and connected, then the maximum eigenvalue $\lambda_1 = k$ occurs with multiplicity 1. Furthermore, the minimum eigenvalue $\lambda_n \geq -k$ with $\lambda_n = -k$ if and only if G is bipartite. If G is bipartite, then $\lambda_n = -k$ also occurs with multiplicity 1.*

In any normal graph theory course, we would have defined bipartite by now. I was hoping to avoid it all together, but it will be good to know a criterion for when $|\lambda_1| = |\lambda_n|$ later (in a more general setting).

Definition 3.2.13. *Let $G = (V, E)$ be a directed or undirected graph. We say G is* **bipartite** *if there is a partition $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, such that $(u, v) \in E$ implies either $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$.*

In other words, no two vertices in V_1 are connected by an edge, and similarly for V_2. For instance, C_n is bipartite if and only if n is even (draw pictures), where as a path graph on n vertices is bipartite for any n. On the other hand, graphs like K_n, star graphs, and wheel graphs are never bipartite. A convenient way to think the bipartite condition in terms of colorings: we can color vertices in V_1 red, and vertices in V_2 green, so we see a bipartite graph G is one with a 2-coloring, i.e., $\chi(G) \leq 2$. (We need to allow the possibility $\chi(G) = 1$ because technically graphs

with no edges are bipartite.) In terms of matrices, bipartite means that one can order the vertices so that the adjacency matrix is a block matrix of the form $\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$.

Proof. Suppose $v_1 = (x_1, \dots, x_n)$ is an eigenvector for $\lambda_1 = k$. As before, scale v_1 to get a maximum coordinate $x_m = 1$ and $|x_i| \leq 1$ for $1 \leq i \leq n$. Let $(y_1, \dots, y_n) = Av_1 = \lambda v_1 = kv_1$ and $A = (a_{ij})$ Then

$$y_m = \sum_{j=1}^{n} a_{mj} x_j = k$$

implies that $x_j = 1$ for all k neighbors j of vertex m. Similarly, $x_i = 1$ for all neighbors of these x_j's, i.e., the vertices of distance 2 from m. Continuing in this manner shows $x_j = 1$ for all j in the connected component of m, which is the whole graph by assumption. Hence v_1 is determined uniquely up to a scalar, which means λ_1 has multiplicity 1. (Since A is diagonalizable, the geometric and algebraic multiplicity of an eigenvalue are the same.)

For the minimum eigenvalue, we saw already that $\lambda_n \geq -k$ in Proposition 3.2.10. So suppose $(x_1, \dots, x_n)$ is an eigenvector with eigenvalue $-k$, again satisfying $x_m = 1$ and $|x_i| \leq 1$ for $1 \leq i \leq n$. Then we have the identity

$$y_m = \sum_{j=1}^{n} a_{mj} x_j = -k,$$

so now we see $x_j = -1$ for all neighbors j of m. The same reasoning shows that if $x_j = -1$, then $x_i = +1$ for any neighbor i of j. Let $V_1 = \{i : 1 \leq i \leq n,\ x_i = +1\}$ and $V_2 = \{j : 1 \leq j \leq n,\ x_j = -1\}$. By connectedness, every vertex of G lies in either V_1 or V_2, and the reasoning above says that no two vertices in V_1 are connected by a (single) edge, and similarly for V_2. Hence G must be bipartite. In addition, this eigenvector is determined uniquely up to scaling, so if G is bipartite, $\lambda = -k$ has multiplicity 1. □

We won't prove this, but another standard fact is that the spectrum $\{\lambda_1, \dots, \lambda_n\}$ of G is symmetric about 0, i.e., $\lambda_j = \lambda_{n-j+1}$ for $1 \leq j \leq \frac{n}{2}$, if and only if G is bipartite.

Now let's prove a result about the diameter.

Proposition 3.2.14. *G connected simple undirected. If G has r distinct eigenvalues, then* $\operatorname{diam}(G) < r$.

Proof. Let $\lambda_1, \dots, \lambda_r$ denote the distinct eigenvalues of G, and m_i the multiplicity of λ_i. Then we may diagonalize A as $D = SAS^{-1}$ where D is a block diagonal matrix of the form

$$D = \begin{pmatrix} \lambda_1 I_{m_1} & & \\ & \ddots & \\ & & \lambda_r I_{m_r} \end{pmatrix}.$$

Consider the polynomial in A given by

$$m(A) = (\lambda_1 I - A)(\lambda_2 I - A) \cdots (\lambda_r I - A).$$

Note that

$$\begin{aligned} m(D) &= (\lambda_1 I - A)(\lambda_2 I - A) \cdots (\lambda_r I - A) \\ &= S(\lambda_1 I - A)S^{-1} S(\lambda_2 I - A)S^{-1} \cdots S(\lambda_r I - A)S^{-1} = Sm(A)S^{-1}. \end{aligned}$$

Each $\lambda_i I - D$ is a block diagonal matrix where the i-th diagonal block is all zeroes, so when we multiply them all together, we see $m(D) = 0$. For example, if $r = 3$, we have

$$m(D) = \begin{pmatrix} 0 \cdot I_{m_1} & & \\ & (\lambda_1 - \lambda_2) I_{m_2} & \\ & & (\lambda_1 - \lambda_3) I_{m_3} \end{pmatrix} \begin{pmatrix} (\lambda_2 - \lambda_1) I_{m_1} & & \\ & 0 \cdot I_{m_2} & \\ & & (\lambda_2 - \lambda_3) I_{m_3} \end{pmatrix} \times \begin{pmatrix} (\lambda_3 - \lambda_1) I_{m_1} & & \\ & (\lambda_3 - \lambda_2) I_{m_2} & \\ & & 0 \cdot I_{m_3} \end{pmatrix} = 0.$$

(Block diagonal matrices multiply like diagonal matrices: $\begin{pmatrix} A_1 & \\ & A_2 \end{pmatrix} \begin{pmatrix} B_1 & \\ & B_2 \end{pmatrix} = \begin{pmatrix} A_1 B_1 & \\ & A_2 B_2 \end{pmatrix}$.)
Consequently, $m(A) = 0$, so we have a nontrivial linear dependence relation among the first r powers of A and $I = A^0$:

$$m(A) = c_r A^r + c_{r-1} A^{r-1} + \cdots + c_1 A + c_0 = 0.$$

(This is nontrivial as $c_r = (-1)^r \neq 0$.)

Now let d be the diameter of G. Say vertices i and j are distance d apart. By Proposition 3.2.1, the (i,j)-th entry of A^d is nonzero but the (i,j) entry of A^ℓ is 0 for all $0 \leq \ell < d$. Hence, any nontrivial linear dependence relation among $I, A, A^2, \ldots, A^{d-1}, A^d$ must not involve A^d. Similarly, there must exist a vertex j' such that $d(i,j') = d - 1$, and therefore the (i,j') entry of $A^\ell = 0$ for all $0 \leq \ell < d-1$, but the (i,j') entry of A^{d-1} is nonzero. Thus any nontrivial linear dependence relation among $I, A, A^2, \ldots, A^{d-1}, A^d$ cannot involve either A^d or A^{d-1}. Continuing in this way, we see there is no nontrivial linear dependence relation among $I, A, A^2, \ldots, A^{d-1}, A^d$. Consequently, $d < r$. □

The polynomial $m(A)$ in the proof is known as the minimal polynomial of A—it is the minimum degree polynomial such that $m(A) = 0$, the $n \times n$ zero matrix.

For instance, K_n has only 2 distinct eigenvalues, so this result says $\operatorname{diam}(K_n) = 1$. Go back to our examples of (undirected) path and cycle graphs and see what this result tells you about their diameters.

Before we move on, let us just mention a few other simple yet interesting results about adjacency spectra.

- The maximum eigenvalue λ_1 is at least the average degree of G.
- We can bound the chromatic number in terms of the maximum and minimum eigenvalues λ_1 and λ_n by
$$1 + \frac{\lambda_1}{|\lambda_n|} \leq \chi(G) \leq 1 + \lambda_1(G).$$
- If G is regular and not complete, then the clique number of G is at most $n - 1$ minus the number of eigenvalues in $(-1, 1)$.

Before we move on to random walks, let us say a word about Laplacian eigenvalues.

Definition 3.2.15. *Let $G = (V, E)$ be a directed or undirected graph on the ordered set $V = \{1, 2, \dots, n\}$. Let A be the adjacency matrix with respect to V and D be the* **degree matrix** *given by $D = \mathrm{diag}(\deg(1), \deg(2), \dots, \deg(n))$. Here $\deg(i)$ denotes the degree of i if G is undirected and the out degree of i if G is directed. The* **Laplacian** *of G (with respect to the ordering on V) is*

$$L = D - A.$$

The **Laplacian eigenvalues** *are the eigenvalues of L, and the* **Laplacian spectrum** $\mathrm{Spec}_L(G)$ *is* $\mathrm{Spec}(L)$.

As with the adjacency matrix, the Laplacian matrix depends upon the ordering of vertices in general, but if L is the Laplacian with respect to an ordered set V and L' is the Laplacian with respect to an ordered set V', then $L' = SLS^{-1}$ for a suitable permutation matrix S. Consequently, the Laplacian spectrum is also an invariant for G, and in some ways it is nicer than the adjacency spectrum.

For instance, we saw some nice results about adjacency spectra of regular graphs. One nice thing here is that the all-one vector is always an eigenvector for regular graphs, and the eigenvalue is the degree. This was because for regular graphs the adjacency matrix has constant row sums.

By definition of the Laplacian matrix, each row sum is zero—the i-th diagonal entry $\deg(i)$ of D is precisely the sum of the entries of the i-th row of A. Consequently the all-one vector $v = (1, 1, \dots, 1)$ is an eigenvector for L with eigenvalue 0. Assuming G is undirected, L is also a symmetric matrix, so it is diagonalizable and has real eigenvalues by the spectral theorem. We write the eigenvalues of L in increasing order as $\lambda_{L,1} \le \lambda_{L,2} \le \cdots \le \lambda_{L,n}$, where as the adjacency eigenvalues are arranged in decreasing order $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$. This is because for regular graphs, the adjacency and Laplacian eigenvalues correspond, but with reverse ordering.

Proposition 3.2.16. *Let G be a k-regular graph. Then $\lambda_{L,i} = k - \lambda_i$ for $1 \le i \le n$, and we have $0 = \lambda_{L,1} \le \lambda_{L,2} \le \cdots \le \lambda_{L,n} \le 2k$.*

Proof. Since G is k-regular, the degree matrix $d = kI$. Thus

$$\begin{aligned} p_L(\lambda) &= \det(\lambda I - L) = \det(\lambda I - (D - A)) = \det(\lambda I - kI + A) \\ &= \det(-((k - \lambda)I - A)) = (-1)^n p_A(k - \lambda). \end{aligned}$$

Since the eigenvalues of A are the roots λ_i of $p_A(\lambda)$, the Laplacian eigenvalues, are the roots of $p_A(k - \lambda)$, which are just the numbers $k - \lambda_i$. It is clear this map $\lambda_i \mapsto k - \lambda_i$ reverses inequalities $\lambda_i \ge \lambda_j$, so the increasing ordering of $\lambda_{L,i}$'s is justified.

The final statement then follows from Proposition 3.2.10. □

In general, it is still the case that 0 is the minimum eigenvalue $\lambda_{L,1}$, so all other (distinct) eigenvalues of L are positive. We also know $\mathrm{tr} L = \sum \deg(i) = \sum \lambda_{L,i}$ is twice the number of edges of G. (This is equal to the number of edges when G is directed, but some eigenvalues might be complex.)

In fact, it's often even better to work with the **normalized Laplacian** $\mathcal{L} = D^{-1/2} L D^{-1/2}$. (Since D is diagonal with nonnegative entries, $D^{-1/2}$ makes sense—just replace each nonzero entry of D with the reciprocal of the square root of that entry. E.g., if $D = \mathrm{diag}(2, 1, 1, 0)$, let $D^{-1/2} = \mathrm{diag}(1/\sqrt{2}, 1, 1, 0)$.) Again all eigenvalues $\lambda_{\mathcal{L},1} \le \cdots \lambda_{\mathcal{L},n}$ of $\mathcal{L}$ are real since $\mathcal{L}$ is still symmetric,

but now they lie in the range $[0, 2]$, with the minimum one being $\lambda_{\mathcal{L},1} = 0$. If G is connected, 0 occurs with multiplicity 1. It turns out that the sizes of the smallest positive eigenvalue $\lambda_{\mathcal{L},2}$ and the largest eigenvalue $\lambda_{\mathcal{L},n}$ give us a lot of information about the network flow, that is not easily accessible from adjacency eigenvalues in general. We will see this by working with the closely related transition matrices in the next section, which have eigenvalues in the range $[-1, 1]$.

Note: you can numerically compute eigenvalues in Sage as with the command `spectrum()`. If you set the optional parameter to `True`, it will return the (non-normalized) Laplacian eigenvalues. You can also compute the characteristic polynomial $p_G(\lambda)$ exactly with the `charpoly()` function. For example:

Sage 6.1

```
sage: G = graphs.CycleGraph(5)
sage: G.charpoly()
x^5 - 5*x^3 + 5*x - 2
sage: G.spectrum()
[2, 0.618033988749895?, 0.618033988749895?, -1.618033988749895?,
-1.618033988749895?]
sage: G.spectrum(True)
[3.618033988749895?, 3.618033988749895?, 1.381966011250106?,
1.381966011250106?, 0]
```

Exercises

Exercise 3.2.1. *Write a function in Sage* `countpaths(G,u,v,l)` *which counts the number of paths from u to v of length l in G by exponentiating the adjacency matrix of G.*

Exercise 3.2.2. *Let G be a k-regular graph. Prove an exact relation between $\mathrm{tr}(A^4)$ and the number of 4-cycles in G.*

Exercise 3.2.3. *Determine (by hand) the spectrum of C_6. Verify Propositions 3.2.10 and 3.2.14 hold for this graph.*

Exercise 3.2.4. *Determine (by hand) the spectrum of the star graph on 5 vertices. Verify Propositions 3.2.10 and 3.2.14 hold for this graph.*

Exercise 3.2.5. *Let G be a connected graph with maximum degree k. Show that k is an eigenvalue of G if and only if G is regular.*

Exercise 3.2.6. *Let G be a directed graph. Show that all eigenvalues of G are 0 if and only if G has no cycles (including cycles of length 1 or 2).*

Exercise 3.2.7. *Suppose G is regular. Express the eigenvalues of the normalized Laplacian $\mathcal{L}$ in terms of the adjacency eigenvalues.*

3.3 Random walks on graphs

▶ Here our graphs may be directed or undirected, simple or non-simple.

Since the works of Erdös, Rényi and Gilbert on random graphs, it was realized that probabilistic methods provide a powerful tool for studying graph theory. One revelation was that you can learn

a lot about the graph by studying random walks. Random walks are simply walks or paths along a graph, where at each stage you choose a direction (edge) to travel along at random. Here is a somewhat more formal definition.

Definition 3.3.1. *A* **random walk** *on a graph $G = (V, E)$ starting at $v_0 \in V$ is a random process which generates an infinite path $(v_0, v_1, v_2, \ldots)$ on G subject to the following rule:*

- *At any time $t = 1, 2, 3, \ldots$, v_t is chosen at random from among the neighbors of v_{t-1}. Each neighbor is equally likely to be chosen. If v_{t-1} has no neighbors, then $v_t = v_{t-1}$.*

Example 3.3.2 (Gambler's Ruin)**.** *For this example only, we will allow G to be infinite. Namely, let $G = (V, E)$ where $V = \mathbb{Z}$ and $E = \{\{i, i+1\} : i \in \mathbb{Z}\}$.*

$$\cdots \text{——} -2 \text{——} -1 \text{——} 0 \text{——} 1 \text{——} 2 \text{——} \cdots$$

Let's consider a random walk on G starting at $v_0 = 0$. Then, we either go left or right with equal probability. Hence at our next step, $t = 1$, we go to $v_1 = 1$ with probability $1/2$ and $v_1 = -1$ with probability $1/2$. We can write this as $P(v_1 = 1) = P(v_1 = -1) = 1/2$. At our the next step, $t = 2$, we see our probabilities depend on what v_1 was. For instance, if $v_1 = 1$, then we are at 0 with probability $1/2$ and 2 with probability $1/2$.

If we want to be systematic, we can list out all possibilities for the walk up to $t = 2$ as

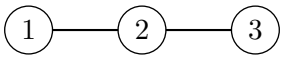
$$(0, -1, -2, \ldots)$$
$$(0, -1, 0, \ldots)$$
$$(0, 1, 0, \ldots)$$
$$(0, 1, 2, \ldots)$$

which must each occur with the same probabilities, namely $1/4$. Hence $P(v_2 = -2) = P(v_2 = 2) = 1/4$ and $P(v_2 = 0) = 1/2$.

This example is called Gambler's Ruin for the following reason. Imagine playing a fair game of chance repeatedly, where your odds of winning are 50–50. Each time you win you gain \$1, and each time you lose, you lose \$1. Then this random walk models your winnings/losings—namely v_t models the number of dollars you have won (or lost if it is negative) after t rounds of the game. One can prove that with probability 1, v_t will get arbitrarily large and arbitrarily small (meaning arbitrarily negative) as $t \to \infty$. (In fact with probability 1, v_t will make large positive and negative swings infinitely often.) Consequently, if you keep playing, you are guaranteed to go bankrupt at some point.

But I say, go for it anyway! It's fun, right! And if you lose all your money, don't get down on yourself—you just verified a mathematical theorem![*]

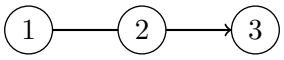
Example 3.3.3. *Suppose G is a path graph on 3 vertices.*



[*]Legal disclaimer: While I may find mathematical satisfaction in your gambling debts, I do not claim any responsibility. Do not tell anyone to break my legs instead of yours.

Take a random walk starting, say, at $v_0 = 1$. Then at the first step, $t = 1$, we have to walk to vertex $v_1 = 2$. At $t = 2$, we go to either vertex $v_2 = 1$ or $v_2 = 3$, with equal probability. Then, regardless of this choice, at $t = 3$, we see we must be back to vertex 2, and therefore this process repeats. Hence we can describe the random walk as $v_{2i+1} = 2$ for all $i \geq 0$ and v_{2i} is either 1 or 3, each with probability $1/2$, for any $i \geq 1$.

We note the condition about v_{t+1} having no neighbors in the definition of random walk is mostly to deal with the situation of nodes of out degree 0 in directed graphs. It can never happen that we reach a node of degree 0 for undirected graphs (unless we start at one). But consider this directed example:



Again, if we start at vertex 1, we must go to vertex 2 next, and from there we either go to 1 or 3. If we go to 1, we must go back to 2 and repeat the process, but if we go to vertex 3, the random walk rule says $v_t = 3$ from this point out. (One could alternatively terminate the random walk here, but this definition will give us a more uniform treatment of random walks.) One can easily check that, as $t \to \infty$, the probability of only bouncing back and forth between vertices 1 and 2 goes to 0—thus with probability 1 we will eventually go to vertex 3, where we will be stuck for eternity. How *ad infinitum*.

As you might expect, these elementary logical arguments for understanding random walks will get cumbersome very quickly with more complicated graphs. There's a better way to look at this problem using probability and linear algebra. In fact, this is a more precise way to think about random walks. At each time t, let us think of the state v_t of the random walk as a *probability distribution* on the graph G (or more precisely, on the set of vertices V). Namely, regard v_t as a *probability vector* (a vector with non-negative entries which sum to 1) $v_t = (p_1(t), p_2(t), \ldots, p_n(t))$, where $p_i(t)$ is the probability that we are at vertex i at time t. (We think of these vectors as column vectors, despite me just having written all the entries in a row.) Then to describe the random walk, we just need to explain how to get from the probability vector v_t to the next one v_{t+1}. This will be done by means of the transition matrix T (which will independent of both our time t and the initial state v_0).

Let's see how this works first by redoing Example 3.3.3. Consider the matrix

$$T = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 1 & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{pmatrix}.$$

Then

$$Tv_t = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 1 & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \end{pmatrix} = \begin{pmatrix} p_2(t)/2 \\ p_1(t) + p_3(t) \\ p_2(t)/2 \end{pmatrix}.$$

First observe that as long as v_t is a probability vector, i.e. $p_1(t) + p_2(t) + p_3(t) = 1$ and all entries are non-negative, then so is Tv_t. Now the key point is that multiplication by T perfectly describes this random walk process: the probability that we are at vertex 1 (or 3) at time $t + 1$ must be $1/2$ the probability we were at vertex 2 at time t, and the probability that we are at vertex 2 at time

$t+1$ must be the probability that we were at either vertex 1 or 3 at time t. Thus we can compute the random walk as

$$v_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad v_1 = Tv_0 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad v_2 = Tv_1 = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}, \quad v_3 = Tv_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = v_1, \quad \dots$$

Here the state vector v_0 means that (with probability 1) we are at vertex 1 at time 0, v_1 means (with probability 1) we are at vertex 2 at time 1, and so on.

Here is the general way to compute this.

Definition 3.3.4. *Let $G = (V, E)$ be a graph (possibly directed and non-simple) with the ordered set of vertices $V = \{1, 2, \dots, n\}$. Let $A = (a_{ij})$ be the adjacency matrix with respect to V. Then the* **transition matrix** *for G with respect to V is the matrix $T = (t_{ij})$ where*

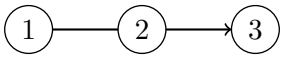
$$t_{ij} = \frac{a_{ji}}{\deg(j)}$$

if $\deg(j) \neq 0$ *and*

$$t_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

if $\deg(j) = 0$. *Here* $\deg(j)$ *denotes the out degree of j in the case G is directed.*

Put more plainly, for undirected graphs $a_{ij} = a_{ji}$, so the transition matrix T is the matrix obtained by dividing each entry a_{ij} of A by the j-th column sum (unless the j-th column is all 0's, in which case we just change the diagonal entry a_{jj} to 1). For directed graphs, we need to first take the transpose of A and then divide by column sums (except for zero columns, where we set the diagonal entry to 1). For instance, for the graph



we have

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad A^T = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad T = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 1 & 0 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix}.$$

Consequently, the matrix T will be a matrix of non-negative entries whose columns all sum to 1. Note that T need not be symmetric even if A is (we computed the transition matrix T for the path graph on 3 vertices above, and it was not symmetric). However if G is a k-regular graph, then we simply have $T = \frac{1}{k}A$, so T is symmetric in this case.

Now let's see why this transition matrix does what we want. Consider a probability vector $v_t = (p_1(t), p_2(t), \dots, p_n(t))$. (Again this is a column vector—you just can't tell because your head is sideways.) Then

$$Tv_t = \begin{pmatrix} \sum_j t_{1j} p_j(t) \\ \sum_j t_{2j} p_j(t) \\ \vdots \\ \sum_j t_{nj} p_j(t) \end{pmatrix}.$$

Again, we can first note that each entry of the vector on the right must be non-negative, and the entries sum to 1:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}p_j(t) = \sum_{j=1}^{n}\left(p_j(t)\sum_{i=1}^{n} t_{ij}\right) = \sum_{j=1}^{n} p_j(t)\cdot 1 = 1, \tag{3.8}$$

as each column of T and the vector v_t does. (Whenever you see a double sum, you should always try to change the order of summation to get what you want. Even if you don't know what you want, interchange the order of summation and whatever you get is undoubtedly what your soul was desperately longing for.)

What about the i-th entry of Tv_t? This is $\sum_j t_{ij}p_j(t)$. The only contributions to this sum are from j with $t_{ij} \neq 0$, i.e., those j which have an edge to i. For such a j, $t_{ij} = \frac{1}{\deg(j)}$ (or $t_{ij} = 1$ if $i = j$ and $\deg(j) = 0$), but this is precisely the probability that we will proceed from vertex j to vertex i. Thus $t_{ij}p_j(t)$ is the probability—STOP THE PRESSES!!! I HAVE JUST REALIZED I'M USING t FOR TOO MANY THINGS—that you go from vertex j at time t to vertex i at time $t+1$. Hence summing this up over all i must yield $p_i(t+1)$, the probability that you went to vertex i (from some vertex) at time $t+1$.

This shows that the transition matrix can always be used to express the state of the random walk. That is, for any state v_t

$$v_{t+1} = Tv.$$

In particular, for any initial state v_0, we see

$$v_1 = Tv_0, \quad v_2 = Tv_1 = T^2v_0, \quad v_3 = Tv_2 = T^3v_0,$$

and in general

$$v_t = T^tv_0.$$

Example 3.3.5. *Consider $G = C_3$ and a random walk starting at vertex 1. Here the transition matrix is*

$$T = \frac{1}{2}A = \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

The first few steps of the random walk (as probability distributions) is then given by

$$v_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad v_1 = Tv_0 = \begin{pmatrix} 0 \\ 0.5 \\ 0.5 \end{pmatrix}, \quad v_2 = Tv_1 = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad v_3 = Tv_2 = \begin{pmatrix} 0.25 \\ 0.375 \\ 0.375 \end{pmatrix}.$$

Continuing in this manner we see

$$v_4 = \begin{pmatrix} 0.375 \\ 0.3125 \\ 0.3125 \end{pmatrix}, \quad v_5 = \begin{pmatrix} 0.3125 \\ 0.34375 \\ 0.34375 \end{pmatrix}, \quad v_6 = \begin{pmatrix} 0.34375 \\ 0.328125 \\ 0.328125 \end{pmatrix}.$$

If we do some more calculations, it will be apparent that $v_t \to (1/3, 1/3, 1/3)$ as $t \to \infty$.

However, there's a more elegant way to see this. We saw in Example 3.2.9 that a basis of eigenvectors for A (and therefore T) is $(1,1,1)$, $(1,-1,0)$, $(1,0,-1)$. These have, respectively, eigenvalues $2,-1,-1$ for A and therefore eigenvalues $1,-\frac{1}{2},-\frac{1}{2}$ for T. Hence if we take

$$S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}, \quad S^{-1} = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix}$$

then

$$D = STS^{-1} = \begin{pmatrix} 1 & & \\ & -\frac{1}{2} & \\ & & -\frac{1}{2} \end{pmatrix}.$$

(Recall T can be diagonalized by taking S the matrix whose columns are a basis of eigenvectors—in this example S happens to be symmetric, so the rows are also eigenvectors, but this is not true in general.) Therefore

$$T^t = S^{-1}D^tS = S^{-1}\begin{pmatrix} 1 & & \\ & (\frac{-1}{2})^t & \\ & & (\frac{-1}{2})^t \end{pmatrix} S.$$

In particular,

$$v_t = T^t v_0 = S^{-1}D^tSv_0 = S^{-1}D^t\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = S^{-1}\begin{pmatrix} 1 \\ (-2)^{-t} \\ (-2)^{-t} \end{pmatrix} = \frac{1}{3}\begin{pmatrix} 1+2(-2)^{-t} \\ 1-(-2)^{-t} \\ 1-(-2)^{-t} \end{pmatrix}.$$

In particular, we see that as $t \to \infty$, $v_t \to (1/3, 1/3, 1/3)$.

Note that from the symmetry of C_3, we see we get the same result no matter where we start the random walk. This means that if we do a random walk, starting anywhere on this graph, after a reasonable number of steps, we are (essentially) as likely to be at any one vertex i as at any other vertex j. Put another way, in any random walk, we will (with proabability 1) be spending an equal amount of time at each vertex.

Contrast this to the case of the path graph on 3 vertices from Example 3.3.3. Even though the distribution for the random walk doesn't "stabilize," i.e. v_t does not have a well-defined limit, we can still say that at any time $t > 0$, we are twice as likely to be at vertex 2 as at vertex 1 or 3 (if we do not know if t is even or odd). This should be fairly evident, but we can formally do this analysis by averaging the v_t's (see Exercise 3.3.3). Said differently, for a random walk on the path graph on 3 vertices (starting at any vertex in fact), we expect to be spending twice as much time at vertex 2 as at vertices 1 or 3. This indicates that vertex 2 is more central than vertices 1 or 3, unlike the case of C_3 where every vertex is equally central.

As you might have guessed now, this idea can be used to define another notion of centrality: eigenvector centrality. To see how this limiting behavior of random walks is related to eigenvectors, suppose $v^* = \lim_{t\to\infty} v_t$ exists. Then

$$Tv^* = T\lim_{t\to\infty} v_t = \lim_{t\to\infty} Tv_t = \lim_{t\to\infty} v_{t+1} = v^*,$$

i.e., any limiting distribution v^* for the random walk (there can be more than one—think about disconnected graphs) must be an eigenvector for T with eigenvalue 1.

Some graphs, like the path graph of length 3, may not have limits for random walks starting at any vertex. This will clearly be the case for any bipartite graph or something like a directed cycle graph. However, one can still often make sense of the percentage of time p_i spent at vertex i in a random walk, and $(p_1, \dots, p_n)$ will then be an eigenvector with eigenvalue 1. We just illustrate this with a couple of examples.

Example 3.3.6. *Consider a star graph G on n vertices—say the hub is vertex n. (For $n = 3$ this is the path graph on 3 vertices, just with a different labeling of vertices.) This is a bipartite graphs. Take a random walk starting at vertex 1, say. I.e., $v_0 = (1, 0, \dots, 0)$. Then at time $t = 1$, we must travel to the hub. From there, we travel to one of the $n - 1$ "satellite" vertices, then back again to the hub, and so forth. Hence, for $t \geq 1$, we have $v_t = (0, \dots, 0, 1)$ if t is odd and $v_t = (1/(n-1), \dots, 1/(n-1), 0)$ if t is even. So on average, we expect to spend $p_n = 1/2$ of the time at the hub, and $p_i = 1/2(n-1)$ percent of our time at any other given vertex $i < n$.*

For the star graph, the transition matrix is

$$T = \begin{pmatrix} 0 & \cdots & 0 & \frac{1}{n-1} \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \frac{1}{n-1} \\ 1 & \cdots & 1 & 0 \end{pmatrix},$$

and one easily sees that

$$v = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} 1/2(n-1) \\ \vdots \\ 1/2(n-1) \\ 1/2 \end{pmatrix}$$

is an eigenvector for T with eigenvalue 1.

Example 3.3.7. *Consider a directed cycle graph G on n vertices, and a random walk starting at $v_0 = 1$. Then the random walk really has no randomness at all—it must be the infinite repeating path $(1, 2, 3, \dots, n, 1, 2, 3, \dots, n, 1, 2, 3, \dots)$. Consequently we spend $p_i = 1/n$ percent of our time at vertex i. Here the transition matrix equals the identity matrix*

$$T = A^T = \begin{pmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & - & \ddots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

and we see

$$v = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} 1/n \\ \vdots \\ 1/n \end{pmatrix}$$

is an eigenvector for T with eigenvalue 1.

Hence it seems reasonable to use the values of an eigenvector with eigenvalue 1 to define a centrality measure. For this we would like to know if 1 is always an eigenvalue, and if so, if the eigenspace is 1-dimensional. The following well-known theorem from linear algebra addresses this (though this is not typically covered in a first course on linear algebra).

Theorem 3.3.8 (Perron–Frobenius)**.** *Let G be a graph with transition matrix T. Then any (real or complex) eigenvalue λ of T satisfies $|\lambda| \leq 1$. In addition, $\lambda = 1$ is an eigenvalue and it has an eigenvector with all positive entries. The Furthermore, if G is undirected and connected, or directed and strongly connected, the eigenvalue $\lambda = 1$ has multiplicity 1.*

For the adjacency matrix A, there is a number $\rho > 0$, called the **spectral radius** *of A, such that ρ is an eigenvalue of A and any λ of A satisfies $|\lambda| \leq \rho$. The eigenvalue ρ has an eigenvector with all positive entries. If G is undirected and connected, or directed and strongly connected, the eigenvalue $\lambda = \rho$ has multiplicity 1.*

Note we have already proved this in the case that G is k-regular, as then $T = \frac{1}{k}A$, so $\mathrm{Spec}(T) = \{\frac{\lambda}{k} : \lambda \in \mathrm{Spec}(A)\}$ (cf. Proposition 3.2.10 and 3.2.12).

Due to time constraints, we won't prove this in general, but we'll just sketch out the ideas for T.

Suppose v is an eigenvector for T with eigenvalue λ, and the sum s of the entries of v is nonzero. We may scale v so $s = 1$. Then the calculation in (3.8) showing that T takes probability vectors to probability vectors means that the sum of the entries of $Tv = \lambda v$, which is λ, must also be 1.

To complete the first part of the theorem, it remains to show $\lambda = 1$ if $s = 0$. Let me just explain what to do if λ and v are real. Then some of entries are positive and some are negative. So we can write $v = v^+ + v^-$ where all entries of v^+ are positive and all entries of v^- are ≤ 0. Now one can use the fact that the sums of entries of Tv^+ and Tv^- are the same as the sums of the entries of v^+ and v^- (this follows as in (3.8)) to deduce that $|\lambda| \leq 1$.

If all eigenvalues $|\lambda| < 1$, then T must shrink the "size" of any v, i.e., $T^t v \to 0$ as $t \to \infty$. However this contradicts the fact that left multiplication by T preserves column sums. Hence we must have some $|\lambda| = 1$, and looking at v's with positive column sums will tell us in fact some $\lambda = 1$.

One can show that any eigenvector v for $\lambda = 1$ must have all nonnegative entries. Hence we can scale v to be a probability vector. Then v must essentially be the limiting distribution of some random walk. If G is (strongly) connected, then in any random walk we will spend a nonzero percentage p_i of our time at vertex i. One can prove then that $v = (p_1, p_2, \ldots, p_n)$.

Note that if G is not connected, e.g., a disjoint union of two copies of the cycle graph C_3, then $v_1 = (1/3, 1/3, 1/3, 0, 0, 0)$ and $v_2 = (0, 0, 0, 1/3, 1/3, 1/3)$ are both eigenvectors with eigenvalue 1. Consequently, so is any linear combination of v_1 and v_2. So $\lambda = 1$ will have multiplicity > 1 (it will be the number of connected components if G is undirected).

Eigenvector centralities

Definition 3.3.9. *Let G be a (strongly) connected graph and $v = (p_1, \ldots, p_n)$ the unique probability vector such that $Tv = v$. Then the* **random walk eigenvector centrality** *of vertex i is p_i.*

In other words, the eigenvector centrality of vertex i is the percentage of the time you spend at vertex i if you just wander around the graph randomly. We can explain how this reflects "centrality" as follows.

Let's think of a social network, like a Twitter graph, or a graph of webpages. Pretend you just start out somewhere random in the graph, and randomly visit other Twitter users or webpages on this network by randomly clicking users followed or webpage links. This is a random walk. The more popular a Twitter user or webpage is, the more it will be followed by or linked to from other nodes in the graph. Consequently, the more often you will visit it on this random walk.

However, this does not just give a measure of degree centrality, because the amount of time you spend at a given node will depend on the amount of time you spend at nodes that link to it. In other words the centrality of a given node is determined by not only the number of nodes linking to it, but by their centrality also. If you only have a few links to your webpage, but they're from Youtube, Wikipedia, the New York Times and Black People Love Us, you will get a lot more traffic than if you are linked to from 100 webpages the Internet has never heard of.

Another way to think of this is that each nodes links in the graph count as a sort of "popularity vote" for the nodes they link to. But these votes aren't democratic or limited—you can vote as many times as you want, however the more votes you make, the less each individual vote is worth. On the other hand, your votes count for a lot more if you personally are popular, i.e., if you got a lot of votes yourself.

For instance, let's say I tweet about graph theory, and my arch-nemesis Steve tweets about differential equations. I only have 3 followers, and Steve has 10, but none of our followers are too popular. Then, one day, Kanye West gets interested in social network analysis and decides to start following me. If Kanye only follows me and a few other people, I will get a lot of kickback traffic from Kanye's followers (though people tracing followers or from Kanye retweeting my brilliant graph theory tidbits), and I may get some more direct followers this way. Thus my popularity will blow Steve's to bits!

However, if Kanye follows 10000 people, almost no one will notice Kanye's following me, and he's not likely to often retweet my pith. So maybe Steve and I will stay at around the same level of popularity, until someone finds out his last name is Hawking, and he becomes super popular because people confuse him with a genius who can't walk. I don't know why everyone thinks Stephen Hawking is so smart anyway. His wheelchair does all the talking for him. Anyone could be a genius with that wheelchair, even Steve! Twitter's stupid, with stupid followers anyway. I don't need Twitter. I can make my own social network—a social network of one—and it won't have followers, just leaders. And no Steves allowed.

The probability vector $(p_1, \dots, p_n)$ with eigenvalue 1 (assuming the graph is stongly connected) can be interpreted as p_i representing the percentage of votes vertex i gets. This is for the following reason. Lets say we will vote in rounds for the centrality of a vertex, and a really large number of ballots can be cast. Start with some initial distribution $(p_1(0), \dots, p_n(0))$ of how important the nodes are—namely $p_i(0)$ represents the percentage of total votes (ballots) vertex i has to start with. At each round of voting, we revise the centrality (number of total votes) vertex v_i by giving i the percentage $p_i(t+1)$ of votes it just received. This is given by the rule

$$\begin{pmatrix} p_1(t+1) \\ \vdots \\ p_n(t) \end{pmatrix} = T \begin{pmatrix} p_1(t) \\ \vdots \\ p_n(t) \end{pmatrix}.$$

If this converges, it must converge to $(p_1, \dots, p_n)$. Even if it does not converge for our intial choice of $(p_1(0), \dots, p_n(0))$, if we *were* to take $(p_1(0), \dots, p_n(0)) = (p_1, \dots, p_n)$, then

$$\begin{pmatrix} p_1(t) \\ \vdots \\ p_n(t) \end{pmatrix} = T^t \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$$

for all t since this is an eigenvector with eigenvalue 1 for T. In other words, with this judicious choice of initial centralities $(p_1, \dots, p_n)$, voting does not change any nodes popularity at any step.

This means $(p_1, \dots, p_n)$ must be a solution to the recursively-defined problem: find a measure of centrality where each the centrality of each node is proportional to the centrality of the nodes linking to (or voting for) it.

This notion of eigenvector centrality is the basis for Google's PageRank algorithm, however we will need to address the issue that hyperlink graphs are not strongly connected.

This is all rather clever, but for undirected graphs, this turns out to be a little too clever, and we might want to modify this measure of centrality (it's still very useful for directed graphs, as evidenced by PageRank). The issue is that the value of node j voting for node i diminishes with the number of nodes vertex j votes for. Suppose we're in an undirected social network. Having a higher degree (e.g., Kanye) should make a node more important. And being connected to important people should make you more important. However, if you're connected to Kanye, but he's connected to 1,000,000 other people, this connection doesn't transfer much influence to you. In fact, this measure simply reduces to a measure of degree centrality. (This is less of an issue for directed graphs, where, say, Kanye probably has a lot more followers than followees.)

So we might like to allow nodes to be able to vote in such a way that their voting power does not get divided. We can do this by just using the adjacency matrix rather than the transition matrix, but now we will need to scale the values since Av does not typically have the same sum as v. Start with some initial probability vector $v_0 = (p_1(0), \dots, p_n(0))$ and iterate this by

$$v_{t+1} = \frac{1}{|Av_t|} A v_t,$$

where $|v|$ denotes the sum of entries in the vector v. This way v_{t+1} will still be a probability vector, and this process does not dilute the votes of nodes of high degree—each of Kanye's votes are worth more than each of mine, which was not true under the random walk process using T. Under some mild hypotheses, $v^* = \lim_{t\to\infty} v_t$ exists, and is an eigenvector for A with eigenvalue ρ, the largest positive eigenvalue. We'll see this below, but let's first state the definition and see a couple of examples.

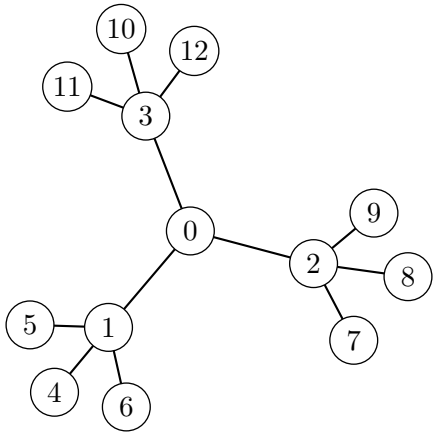
Definition 3.3.10. *Let G be a connected undirected graph with adjacency matrix A. Say ρ is the maximum positive eigenvalue of A, and $v = (p_1, \dots, p_n)$ is a probability vector which is an eigenvector for eigenvalue ρ. Then p_i is the* **(adjacency) eigenvector centrality** *of vertex i.*

The Perron–Frobenius theorem guarantees the existence and uniqueness of such a v. One can similarly define adjacency eigenvector centrality for directed graphs, but then it makes more sense to use the transpose A^T. Some authors normalize the eigenvector v so that the maximum entry is 1.

Example 3.3.11. *Let G be a k-regular graph. Then each vertex has both random walk and adjacency eigenvector centrality $\frac{1}{n}$. This is because $v = (1/n, \dots, 1/n)$ is an eigenvector for A, whence for $T = \frac{1}{k}A$.*

So for regular graphs, no nodes are more central than any other nodes—that is in any random walk we will *eventually* spend an equal amount of time at any vertex. This will happen sooner, say for complete graphs than for cycle graphs, and we will come back to this point as a way to measure how good network flow is.

Example 3.3.12. *Consider the tree G given by*



By iterating a random walk (and averaging), we can numerically find an eigenvector $v = (p_0, \dots, p_{12})$ *of* T *with eigenvalue 1 given by*

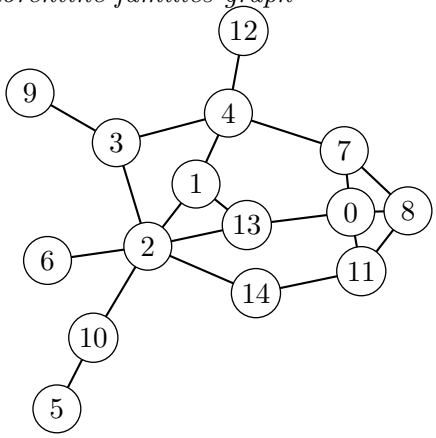
$$p_i = \begin{cases} \frac{1}{8} & i = 0 \\ \frac{1}{6} & i = 1, 2, 3 \\ \frac{1}{24} & else \end{cases}$$

and verify it is an eigenvector by matrix multiplication. In other words, the random walk eigenvector centrality of vertex i *is just* $\frac{deg(i)}{24}$*. Note* 24 *is twice the size of* G*, i.e., the sum of all vertex degrees.*

By iterating the process $v \mapsto \frac{1}{|Av|}Av$ *(and averaging), we can numerically find an eigenvector* $v = (p_0, \dots, p_{12})$ *(which again we can verify afterwards) with maximium eigenvalue* $\rho = \sqrt{6}$ *to get the eigenvector centralities*

$$p_i = \begin{cases} \frac{1}{10} & i = 0 \\ \frac{1}{5} & i = 1, 2, 3 \\ \frac{1}{30} & else. \end{cases}$$

Example 3.3.13. *Recall the Florentine families graph*



By iterating a random walk, we numerically compute the random walk eigenvector centralities in Sage as

```
[ 0.10000000000?]
[ 0.07500000000?]
[ 0.15000000000?]
[ 0.07500000000?]
```

```
[ 0.10000000000?]
[0.025000000000?]
[0.025000000000?]
[ 0.07500000000?]
[ 0.07500000000?]
[0.025000000000?]
[ 0.05000000000?]
[ 0.07500000000?]
[0.025000000000?]
[ 0.07500000000?]
[ 0.05000000000?]
```

Again, these numbers are simply the degrees of the vertices divided by the sum of all vertex degrees.

Similarly, by iterating $v \mapsto \frac{1}{|Av|}Av$*, we can estimate the adjacency eigenvector centralities in Sage as*

```
[ 0.1016327190110338?]
[ 0.0930282524780531?]
[ 0.1228533246839824?]
[ 0.0696496754352986?]
[ 0.0825427479718137?]
[0.0127942738390679 4?]
[0.0377301628845139 9?]
[ 0.0807396638222649?]
[ 0.0787212547936576?]
[0.0213904963971233 4?]
[ 0.0416594829674572?]
[ 0.0739521897453467?]
[ 0.0253501590935615?]
[ 0.0975135686688800?]
[ 0.0604420282079456?]
```

These are much more interesting. We see the most central nodes are the vertex 2 and vertex 0. We also note that vertex 1 (0.093...) has considerably higher eigenvector centrality than vertex 11 (0.073...), even though they both have degree 3, because the neighbors of 13 are more central than the neighbors of 11.

Computing stable eigenvectors

Now let's consider the problem of computing random walk and eigenvector centralities. We already seen that we can sometimes do this by looking at limits of random walks. Let explain this in more detail. Suppose G is graph with transition matrix T. Let $\{\lambda_1, \dots, \lambda_n\}$ be the (possibly complex) eigenvalues of T (with multiplicity). Assume for simplicity that (at least over $\mathbb{C}$), we have a basis of eigenvectors $v_1, \dots, v_n$ for T. (This is always the case if G is T-regular, and is true "most" of the time. When it is not, the conclusions we we make will still be valid, but one needs to modify the arguments below—e.g., by putting T in upper triangular form—see Exercise 3.3.4 for the case of $n = 3$.) Hence any $v \in \mathbb{C}^n$ can be written as a linear combination

$$v_0 = c_1 v_1 + \cdots + c_n v_n.$$

Then

$$T^t v_0 = c_1 T^t v_1 + \cdots + c_n T^t v_n = c_1 \lambda_1^t v_1 + \cdots + c_n \lambda_n^t v_n. \tag{3.9}$$

Since each $|\lambda_i| \leq 1$, these terms are bounded, and in fact $\lambda_i^t \to 0$ if $|\lambda_i| < 1$.

We know 1 is an eigenvalue of T—say $\lambda_1 = 1$. If all other eigenvalues are strictly less than 1 in absolute value (real or complex), then we in fact have

$$\lim_{t\to\infty} T^t v_0 = \lim_{t\to\infty} (c_1 \lambda_1^t v_1 + \cdots + c_n \lambda_n^t v_n) = c_1 v_1.$$

If v is a probability vector then, using the fact that multiplication by T preserves column sums, we must have $c_1 = 1$ if we scale v_1 so it's sum is 1. (The above limit implies that we can scale v_1 to get a probability vector.) In other words, $|\lambda_i| < 1$ for all $2 \leq i \leq n$, then $T^t v_0 \to v_1$ for *any* initial state probability vector v_0. That is, no matter where you start the random walk, the long term behavior is exactly the same. Furthermore, this method is typically very fast, even for really large networks, so iterating $T^t v_0$ allows us to numerically compute v_1 (and thus eigenvector centralities) very quickly. In this case v_1 is called a *dominant* or *principal eigenvector* because it dominates the limiting behavior of the random walk.* The existence of a dominant eigenvector is what make random walks an extremely useful tool for analyzing networks.

If the eigenvalue $\lambda_1 = 1$ occurs with multiplicity $m > 1$, say $\lambda_1 = \cdots = \lambda_m = 1$ but $|\lambda_i| < 1$ for $i > m$, then it is still true that $T^t v_0$ always has a limit, but it is no longer uniquely determined. Namely,

$$T^t v_0 = c_1 v_1 + \cdots + c_m v_m.$$

Hence the behavior of random walks can depend upon the initial starting position. The Perron–Frobenius theorem says this doesn't happen for (strongly) connected graphs. In fact the multiplicity m of $\lambda = 1$ will, for undirected graphs, be the number of connected components of G, and one obviously will get a different limit for random walks starting in different components.

Now assume G is (strongly) connected, so $\lambda_1 = 1$ occurs with multiplicity 1. Then the only issue in investigating these limits is that there may be other eigenvalues of absolute value 1. First, suppose for simplicity that G is undirected, in which case all eigenvalues of T are real. Then any other eigenvalue of absolute value 1 is -1—this happens if G is bipartite, but only with multiplicity 1. Assume the eigenvalues are ordered so that $\lambda_1 = -1$. Then we have

$$T^t v_0 \approx c_1 v_1 + (-1)^t c_n v_n$$

for large t as all other eigenvalue powers $\lambda_i^t \to 0$. This only converges if $c_n = 0$, otherwise it oscillates between 2 values: $c_1 v_1 + c_n v_n$ and $c_1 v_1 - c_n v_n$. However, averaging these values gives $c_1 v_1$, and if v_1 is scaled to have sum 1, then $c_1 = 1$ if v_0 is a probability vector. Thus we can still quickly numerically evaluate the unique normalized (so the sum of entries is 1) eigenvector v_1 of multiplicity 1 for bipartite graphs.

Now suppose G is directed. Then there could be many eigenvalues of absolute value 1. Recall the directed cycle graph from 3.2.6. Here $T = A$ and all eigenvalues have absolute value 1—e.g., for $n = 4$ they are ± 1 and $\pm i$. However it is the case that any eigenvalue of absolute value 1 must be a root of unity, i.e., $\lambda^m = 1$ for some m. This means there is a single m such that $\lambda^m = 1$ for all λ of absolute value 1. Consequently, when we try to take the limit of $T^t v_0$ it will (approximately, for t large) oscillate periodically amongst a finite number of states. Again, one can numerically recover the unique normalized eigenvector by averaging these periodic states.

*This is the basis of an important technique is statistics and financial mathematics called principal component analysis.

PageRank

The original version of Google's PageRank algorithm was published in a paper titled "The PageRank Citation Ranking: Bringing Order to the Web" in 1999 by Larry Page, Sergey Brin, Rajeev Motwani and Terry Winograd. It was originally designed as a way to find the most significant webpages linking to your webpage.

To do a web search, first one "crawls" the web. This is done by following links from known pages to find new ones, and indexing each webpage. (And in olden days, people and business manually submitted their pages for categorized listings in Yahoo like an online Yellow Pages. Let me explain—in historic times, there were things called books. They were made out of primitive substances known as ink and paper. Papers were thin rectangles cut out of trees. Ink is a liquid that comes out of sea creatures private parts, and I think also elephants in India. This ink gets sprayed on the paper in the shape of words, and then you glue a bunch of paper together to get something called a book. Glue is a sticky substance made out of dead horses that children squirt in their noses to keep their brains from falling out. Books were what contained all the information we knew before the Internet. A phone book was a kind of book, not made out of cell phones as one might first think, but a book with a list of phone numbers of every person and business in your village. The Yellow Pages were the part of the phone book where business were listed by categories—e.g., Blacksmiths or Cobblers. No one knows how these categories were arranged.)

Once a search engine has a reasonable index of the web, it can accept searches from users, compare those searches to the pages it knows, and return a list of the ones it thinks most relevant, hopefully with pages your more likely to be interested in at the top of the list. Prior to Google, search engines (e.g., Altavista and Lycos) would just compare the text of the webpages indexed to the text you entered in the search box. These would return results in an ordering based on similarities of the text entered and text on the webpage (e.g., if all of your search words were in the title or at the top of the page, or if your search words appears on the page many time or close to each other and in order, the webpage gets ranked higher).

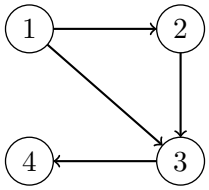
The idea behind PageRank is what the authors called the "random surfer model." It is a variant of the notion of a random walk. Consider a (directed) hyperlink graph. We want to model a "random surf" of the internet. In a random walk, we will start with a random webpage, and randomly visit pages by randomly clicking on links. There are a couple of issues with this. One, we might get stuck on a page with no links. This we can resolve by picking another page at random and starting over again. The more serious issue is that, as our graph is almost surely not strongly connected, we will get stuck in a small portion of the web. This means we won't converge to a meaningful distribution (as in random walk eigenvector centrality) on the nodes—it will be highly dependent on not just our starting node, but the actual choices we make in our random walk as well. Again, this can be resolve by randomly restarting our walk from time to time.

This is all accomplished with the following elegant conceptual idea. When people actually surf the web, they don't search forever (well, maybe in some sense they do, but they don't follow links without restarting forever). People sometimes click on links to continue visiting pages, but sometimes they will start a new "surfing session" at another page. Let's say α is the "restart probability"—the probability that a person starts a new surfing session at any given time. So $1-\alpha$ is the "click-through" probability—the probability of clicking on some link on a given page (i.e., continuing your surfing session).

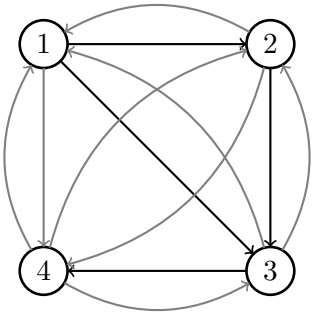
Our default restart probability will be $\alpha = 0.15$, so the click-through probability is $1 - \alpha = 0.85$. This is the value Google originally used—see the 1998 paper "The anatomy of a large-scale

hypertextual Web search engine" by Brin and Page, where they explain their original version of the Google search engine." There $1 - \alpha = 0.85$ is called the damping factor d.

Note we can model this random surfing with restarts as a random walk on a *weighted* directed graph. Here the weights will denote relative probabilities of taking an edge. For instance, let's consider the directed graph



Now we form a complete weighted graph, where the new edges added are in gray.



At each step, with probability $1 - \alpha$, we will travel along a black edge. With probability α, we randomly restart, which means with probability α we take any edge. (Technically, I should have included loops, because there is also the possibility we restart at the vertex we are already at.)

Definition 3.3.14. *Let J denote the all 1 matrix. The* **PageRank matrix** *R_α for G with restart probability α is*

$$R_\alpha = (1 - \alpha)T + \frac{\alpha}{n}J,$$

where T is the transition matrix.

Note the matrix $\frac{1}{n}J$ is the transition matrix for the complete graph K_n with a loop at each vertex. That is, using $\frac{1}{n}J$ for a random walk just means we pick a new vertex at random at each time t—there is no dependence on where we were at time $t - 1$—the probability of being at vertex i at time t is $\frac{1}{n}$ for all i and all t. So R_α is a weighted combination two random processes: (i) doing a random walk on G and (ii) just picking a node completely at random at each step. We do (i) with probability $1 - \alpha$, and (ii) with probability α.

Example 3.3.15. *For the directed graph on 4 vertices pictured above, we have*

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

so with $\alpha = 0.15$,

$$R_\alpha = (1-\alpha)T + \frac{\alpha}{4}J = \frac{1}{4}\begin{pmatrix} 0.15 & 0.15 & 0.15 & 0.15 \\ 1.85 & 0.15 & 0.15 & 0.15 \\ 1.85 & 3.55 & 0.15 & 0.15 \\ 0.15 & 0.15 & 3.55 & 3.55 \end{pmatrix}.$$

One can think of each (j,i) *entry of this matrix as the weight of the edge (or loop if* $i = j$*) from vertex* i *to* j*.*

One key advantage of working with R_α is that now all entries are positive (for $\alpha > 0$). We stated earlier the Perron–Frobenius theorem in our context—this was the version for matrices with non-negative entries. However the Perron–Frobenius theorem for matrices with only positive entries is actually quite a bit stronger—in our case it says the following.

Theorem 3.3.16 (Perron–Frobenius, positive matrix version)**.** *Let* G *be any graph (directed or not, simple or not, connected or not), and* $0 < \alpha \leq 1$*. The PageRank matrix* R_α *has one eigenvalue 1, and all other eigenvalues* λ *satisfy* $|\lambda| < 1$*. Furthermore,* $\lambda = 1$ *has an eigenvector with all positive entries.*

This means there is a unique probability vector $v*$ for such that $R_\alpha v^* = v^*$.

Definition 3.3.17. *Let* G *be any graph on* $V = \{1, \dots, n\}$ *and* $0 < \alpha \leq 1$*, and* $v^* = (p_1, \dots, p_n)$ *the probability vector such that* $R_\alpha v^* = v^*$*. Then the* **PageRank centrality** *(with respect to* α*) of vertex* i *is* p_i*.*

In other words, the PageRank matrix lets us define a centrality measure analogous to random walk eigenvector centrality for *any* graph, not just strongly connected ones.

For Google, what is key is that v^* can be computed efficiently. Since every other eigenvalue $|\lambda| < 1$ by this stronger version of Perron–Frobenius, as we argued above, v^* is a *dominant* eigenvector, in the sense that

$$v^* = \lim_{t \to \infty} R_\alpha^t v_0$$

for any initial state probability vector v_0.

Algorithm 3.3.18. (PageRank)

1. *Start with the initial state* $v_0 = (1/n, \dots, 1/n)$ *and a threshold* $\epsilon > 0$*.*
2. *Compute* $R_\alpha^t v_0$ *successively for* $t = 1, 2, \dots$ *until each entries do not change more than* ϵ*. Record the result as the (approximate) eigenvector* v^**.*
3. *Output* v^* *as the vector of (approximate) PageRank centralities.*

Some remarks are in order:

- In practice, for large graphs, one does not literally need to represent R_α as a matrix to do these calculations. Brin and Page originally considered 24 million webpages and Google indexes many many more now. Instead, one does the calculation with adjacency lists.

- Since the web graph is *sparse*, in that the number of edges grows roughly linearly in the number of nodes. (With about 24 million webpages, there were around 500 million links.) As long as α is not too small, the convergence of $R_\alpha^t v_0$ to v^* is quite fast. Google could take the maximum t to be between 50 to 100. (We'll discuss rate of converge of random walks in the next section.) In 1998, Google could perform do their PageRank in a few hours on a personal computer (and of course faster with more advanced hardware). Note: Google does not need to recompute PageRanks all the time, but just recompute them periodically as they update their index of the web and store the result for use in their search algorithm.

- There are some technical differences with actual original PageRank I have suppressed. For example, if one considers dynamic webpages (e.g., a Google search page), there is no limit to the possible number of webpages. These dynamic pages are essentially ignored when indexing the web. Also, Google dealt with nodes with out-degree 0 ("dangling links") separately. The actual version of PageRank used by Google constantly gets tinkered with to combat people trying to game the system, and the details of the version of PageRank actually in use are private (to make it harder for people to game the system). Companies created vast "farms" of essentially contentless webpages with links to businesses who paid money to these companies to get them increased web traffic, but now this strategy is not so effective. For instance, I heard that at some point (and perhaps still), getting a link from a page which was deemed unrelated to your page would actually lessen your PageRank score.

- PageRank is just one component of Google's search algorithm. In the original version, Google would index the words on each page and count their occurrence. In addition, Google would take in to account "anchor text"—the words on the pages linking to a given page (this allows one to meaningfully index pages without much text—e.g., images or collections of data—and in general provides better context for each webpage). Based on this, Google would give each page a similarlity ranking to your search query, and combine this ranking with the PageRank to give you an ordered set of results. Google started outperforming the popular search engines like Altavista right away (verified by the fact that you probably never heard Altavista, though I used to use it a lot). Note that in practice, one does not need to compute these similarity rankings for all webpages dynamically—Google can get away first with just checking the webpages with high PageRanks against your query, and also storing results for common queries. (I do not know how this is actually implemented, these are just my speculations on how Google returns search results so quickly—I'm fairly certain Google must implement some form of the first speedup I suggest, but I don't know about the second.)

- The term "link analysis" is used for web search algorithms that use the structure of the network (i.e., what pages link to what pages). Google seemed to be the first use of link analysis for web searching, but there are other algorithms. For example, around the same time another link analysis algorithm called HITS by Jon Kleinberg was proposed, where the idea is to classify webpages on a given topic as either hubs (jumping points for more information) or authorities (providing detailed information). This algorithm is the basis for the search engine Ask.com.

- PageRank can be used for ranking other things as well—e.g., sports teams within a conference. One forms a win–loss matrix and applies PageRank to the associated graph. One can vary this by weighting wins with appropriated factors—for instance, by score differentials or how

recent the win was. These algorithms have done quite well in betting tournaments, such as March Madness.

Exercises

Exercise 3.3.1. *In Example 3.3.2, determine the set of possibilities for v_t for $t \geq 1$. When $t = 3$, compute the probability of each of these vertices.*

Exercise 3.3.2. *Fix $m \geq 0$. In Example 3.3.2, prove that the probability v_t lies in the fixed finite interval $[-m, m]$ goes to 0 as $t \to \infty$.*

Exercise 3.3.3. *For a random walk $(v_0, v_1, v_2, \ldots)$, let $\bar{v}_t = (v_0 + v_1 + \cdots + v_t)/(t+1)$. Compute the limit $\lim_{t\to\infty} \bar{v}_t$ (and show it exists) in the following cases (the answer does not depend on which vertex you start at, but you may start at vertex 1 for simplicity):*

(i) G is the path graph on 3 vertices;

(ii) $G = C_4$;

(iii) G is a star graph on n vertices (say vertex n is the hub)

Exercise 3.3.4. *Let T be a 3×3 matrix with column sums 1. Suppose $T = S^{-1}BS$ for 3×3 (possibly complex) matrices B, S where B is of the form*

$$B = \begin{pmatrix} 1 & a & b \\ & \lambda & c \\ & & \lambda \end{pmatrix},$$

and $|\lambda| < 1$.

(i) For any vector $v \in \mathbb{C}^3$ show $v^* = \lim_{t\to\infty} B^t v$ exists and satisfies $Bv^* = v^*$.

(ii) Deduce that for any probability vector v_0, $\lim_{t\to\infty} T^t v_0$ exists and equals the unique probability vector v^* such that $Tv^* = v^*$.

3.4 The spectral gap and network flow

In computing random walk/adjancency eigenvector or PageRank centralities, we approximated a dominant eigenvector by looking at the convergence of some kind of random walk. This is only an efficient procedure if the convergence is fast.

Let's think about two simple examples: C_n and K_n. Take a random walk starting at vertex 1. Then, as long as n is odd for C_n (otherwise C_n is bipartite), the random walk converges to the "flat" distribution $(1/n, \ldots, 1/n)$. How fast will these things converge?

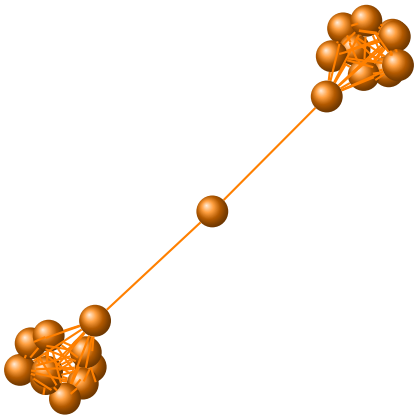
For C_n, it is not even possible to get to every vertex before time $t = (n-1)/2$. However at this point, you are much more likely to still be closer to vertex 1 than one of the vertices diametrically opposite it. So it will take even more time for this distribution to even out.

On the other hand, for K_n, already at $t = 1$ you are equally likely at any vertex besides 1, i.e., each vertex besides 1 had probability $1/(n-1)$. At $t = 2$, the probability of going back to 1 is simply $1/(n-1)$. For any other vertex i, the probability of being there at $t = 2$ is the probability you weren't there at $t = 1$ (which is $1 - 1/(n-1)$) times $1/(n-1)$, i.e., $1/(n-1) - 1/(n-1)^2$. Thus, already at $t = 2$ all of these probabilities are close to the limiting probability of $1/n$, at least for n reasonably large.

In other words, the time required for convergence (say for each coordinate to get within some threshold $\epsilon > 0$ of the limiting value) grows (at least) linearly for C_n as $n \to \infty$, whereas it is bounded for K_n. (In fact, for K_n, convergence is *faster* for larger values n.) Remark: the fast convergence for K_n is what ensures the PageRank algorithm converges quickly if α is not too small—since with probability α we are essentially doing a random walk on K_n (with loops).

We can think of the meaning of this time required for convergence as the amount of time required for a random walk to even out—or, put another way, the time required for the random walk to "forget" it's initial state. The random walk will even out faster, the faster you can get from the initial vertex to every other vertex in the graph. However, the speed of converges is not just determined by distance/diameter. In order for this convergence to be fast, there should be many short ways to get from one vertex to another.

For instance, consider a graph G on $2n+1$ vertices formed as follows: take two disjoint copies of K_n's and a vertex v_0, and connect v_0 to each K_n with a single link.



This graph has diameter 4, no matter how large n is. Now do a random walk, say starting at some vertex in the first copy of K_n. If n is large, it will take a long time to get to the "bridge vertex" v_0. First one has to get to the vertex in K_n connected to v_0, which on average happens one every n steps. From this vertex, the chance of going to v_0 is $1/n$, so we expect it will take about n^2 steps to get to the hub. From the hub, it will now go to each copy of K_n with equal probability. But the point is this argument shows it will take on the order of n^2 steps for the random walk to get reasonably close to the limiting distribution.

The issue is that if there aren't many paths from one part of your graph to another, then it will take a long time for a random walk to spread out evenly over these parts. In fact, one can also do a similar argument with a diameter 2 graph—connect v_0 to every other vertex in the example above. Then a similar argument says it will take on the order of n steps for a random walk starting in one of the K_n's to approach it's limit. In this graph, there are in some sense many paths from one K_n to the other, but they all travel through the "hub" v_0, and the chances of getting to the hub are small (around $1/n$) at each step. So a better conclusion to draw is that if there're aren't many *disjoint paths* from one part of a graph to another, then the random walk will be slow to converge.

Turning this idea around, we see that if the random walk converges quickly, there should be many disjoint ways to get from any one part of the graph to any other part of the graph. In addition, these should also be short paths. (Think of the cycle graph. Or, in our example of 2 K_n's connected by v_0, think about extending the distance between the K_n's by inserting many extra vertices on the links from each K_n to v_0. Now it will take much longer to get to v_0—this time is not just extended by the number of vertices added to this path because with high probability there

will be considerable backtracking, and possibly one will first return to the K_n one came from. Cf. Exercise 3.4.1.)

These were precisely the criteria we wanted our communication/transportation networks to have—having small diameter means the network is efficient. Having many short paths between two points means that the network is still efficient even if some nodes/links fail (which implies good vertex/edge connectivity), and rerouting can be done to handle traffic issues. In short, we can summarize this as saying: (strongly/connected) networks where random walks converge quickly have good *network flow*.

Now let's think about we can measure this notion of network flow—in other words, the rate of convergence of the random walk—more precisely. For simplicity, let us assume we are working with a graph G such that the transition matrix T has a basis of eigenvectors $\{v_1, \dots, v_n\}$ and v_1 is a dominant eigenvector v_1—i.e., if $\lambda_1, \dots, \lambda_n$ are the eigenvalues, then $\lambda_1 = 1$ and $|\lambda_i| < 1$ for all $i \geq 2$. This happens "most of the time" (with high probability for random (strongly) connected graphs). In particular, we know it happens for regular, non-bipartite graphs (cf. Section 3.2). Let v_0 be the initial state vector for a random walk. Write

$$v_0 = c_1 v_1 + \cdots c_n v_n.$$

Then

$$v_t = T^t v_0 = c_1 v_1 + c_2 \lambda_2^t v_2 + \cdots + c_n \lambda_n^t v_n.$$

Writing $v^* = c_1 v_1$ (the limiting value of v_t's), we know

$$\lim_{t \to \infty} (v_t - v*) = \lim_{t \to \infty} c_2 \lambda_2^t v_2 + \cdots + c_n \lambda_n^t v_n = 0. \tag{3.10}$$

Asking for how fast v_t converges to v^* is the same as asking how fast this quantity goes to 0. This is equivalent to asking the rate at which the λ_i^t's go to 0 (for $i \geq 2$). This is simply determined by $|\lambda_i$. The smaller each $|\lambda_i$ is, the faster the term $c_i \lambda_i^t v_i \to 0$. If we order the λ_i's so that $1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \geq 0$, then the rate at which the expression in (3.10) goes to zero is determined by the most dominant term, $c_2 \lambda_2^t v_2$, i.e., by $|\lambda_2|$. (It might be that $|\lambda_i| = |\lambda_2|$ for some other i, but then term i still go to zero at the same rate as term 2, and all other terms go to zero faster, so it suffices to look at $|\lambda_2|$. Also note that ordering of eigenvalues is not necessarily unique, but this again is not important for us.) Hence the smaller the second largest (in absolute value) eigenvalue $|\lambda_2|$ is, the faster the random walk converges, and the better the network flow.

Definition 3.4.1. *Let G be a graph (directed or undirected, weighted or unweighted, simple or not) with transition matrix T. Order the eigenvalues of T so that $1 = \lambda_1 \geq |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \geq 0$. The number $1 - |\lambda_2|$ the* **(random walk) spectral gap**.

The quantity $1 - |\lambda_2|$ is called the spectral gap, as it is the size of the gap between the largest and second largest (absolute value of) eigenvalues of T, and from the point of network flow/random walks, λ_2 is the most important eigenvalue. Note if the spectral gap is 0, i.e., $|\lambda_2| = 1$ (which happens, say, if G is not (strongly) connected or if G is bipartite) this means a typical random walk will not converge to a unique distribution. As long as the spectral gap is > 0, i.e., $|\lambda_2| < 1$, then the random walk converges to a unique dominant eigenvector, and the spectral gap measure the rate of convergence—the larger the spectral gap (i.e., the smaller $|\lambda_2|$), the better the network flow.

Example 3.4.2. *Let $G = K_n$. From Example 3.2.9, we know that the eigenvalues of $T = \frac{1}{n-1}A$ are $\{1, -\frac{1}{n-1}, \dots, -\frac{1}{n-1}\}$. Hence the spectral gap for K_n is $1 - |\frac{1}{n-1}| = \frac{n-2}{n-1}$. This is large (getting closer to 1 for large n), and of course this has the best network flow possible for simple, undirected graphs.*

Example 3.4.3. *Let $G = C_n$. From Example 3.2.8, we know the second largest (absolute) eigenvalue of $T = \frac{1}{2}A$ is $|\lambda_2| = 1$ if n is even and $|\lambda_2| = \cos(2\pi/n)$ if n is odd. Hence the spectral gap is 0 when n is even or $1 - \cos(2\pi/n)$, which is close to 0 for n large, when n is odd. This agrees with our observations that small spectral gap should mean poor network flow.*

Thus we can compare the network flow in different graphs by looking at the spectral gap (for large graphs, one way to estimate is by looking at how fast random walks converge). But if we want to actually construct a network with as good of a flow as possible, it would be nice to have a good theoretical upper bound on the spectral gap. One is at least known in the case of regular graphs.

Theorem 3.4.4 (Alon–Bopanna)**.** *The spectral gap for* any *k-regular graph G on n nodes satisfies*

$$|\lambda_2| \geq \frac{2\sqrt{k-1}}{k} - \epsilon(n,k)$$

i.e., the spectral gap

$$gap(G) \leq 1 - \frac{2\sqrt{k-1}}{k} + \epsilon(n,k)$$

where for a fixed k, the error $\epsilon(n,k) \to 0$ as $n \to \infty$.

Fix $k \geq 2$. In other words, for n large, the best we can do for the spectral gap is about $1 - \frac{2\sqrt{k-1}}{k}$. (In fact a more precise bound is given by Nilli involving the diameter of G.) This motivates the following definition

Definition 3.4.5. *We say a k-regular graph G on n nodes is* **Ramanujan** *if the second largest (absolute) eigenvalue λ_2 satisfies*

$$|\lambda_2| \leq \frac{2\sqrt{k-1}}{k}$$

i.e., if the spectral gap is

$$gap(G) \geq 1 - \frac{2\sqrt{k-1}}{k}.$$

The Alon–Bopanna theorem says that this is optimal for n large. This terminology was introduced by Lubotzky–Phillips–Sarnak in 1988, who explicitly constructed infinite families of Ramanujan graphs for $k = p+1$ where p is a prime of the form $4x+1$ by making use of the something in number theory known as the Ramanujan Conjecture (which has been already proven). Their construction is somewhat technical so we will not describe it here. Since, other constructions has been made, but it is not known if infinitely many Ramanujan graphs exist for all k. (When $k = 2$, the bound above is simply that $|\lambda_2| \geq 0$, which is automatic, so cycle graphs, and unions thereof—these are the only 2-regular graphs—are automatically Ramanujan.)

One might wonder if random regular graphs are likely to be Ramanujan, or "almost Ramanujan" in the sense that the spectral gap is not too far from optimal. This seems reasonable from the point

of view of the Alon–Bopanna theorem, which says that as n gets large, k-regular graphs get closer and closer to being Ramanujan. Indeed, Alon conjectured that "most" random regular graphs are almost Ramanujan (for any $\epsilon > 0$, most random k-regular graphs satisfy $|\lambda_2| \leq \frac{2\sqrt{k-1}}{k} + \epsilon$), and this has been proved by Friedman.

Let me just mention another notion of graphs which make good networks and has been well studied in mathematics and computer science. This is the notion of **expander graphs** (whose definition makes sense for non-regular graphs as well). Technically, individual graphs are not defined to be expander graphs, but a family of graphs is. (Similar to the way it doesn't make sense to define a single graph to be a random graph.) I won't give the precise definition, but the idea is that an expander family is a sequence of graphs G_n with increasing orders n (these graphs need not be subgraphs of each other) such that the network flow is good as $n \to \infty$. Here the notion of network flow is defined by a geometric parameter, called expansion, rather than a spectral one in terms of eigenvalues. Roughly, the expansion parameter measures the percentage of nodes that are neighbors of vertices in S as S ranges over different subsets of vertices. For instance, suppose you know that for any pair of vertices in G, the union of their neighbors contains ever other vertex in the graph. This implies the diameter is at most 2. A consequence of the definition is that in a sequence of expanders G_n, the number of edges grows at most linearly in n while the diameter grows very slowly—logarithmically in n. In other words, the networks maintain low cost yet are efficient.

Here is the first explicit construction of expanders.

Example 3.4.6 (Margulis, 1973)**.** *Let $V_n = \{(i,j) : 0 \leq i,j < n\}$. Define G_n on V_n as by the rule: the neighbors of (i,j) are defined to be $(i+j,j)$, $(i-j,j)$, $(i,i+j)$, $(i,i-j)$, $(i+j+1,j)$, $(i-j+1,j)$, $(i,i+j+1)$, $(i,j-i+1)$. Here these operations are taken mod n so the results always lie between 0 and $n-1$—e.g., if $i+j > n$ we subtract n, or if $i-j < 0$ we add n. This gives us a family G_n of 8-regular graphs, which is a family of expander graphs.*

If a family of graphs G_n has spectral gaps that don't get too small (e.g., don't go to 0), it will be a family of expanders. In particular, any sequence of Ramanujan graphs (with growing orders) will be a sequence of expanders. However it is possible that in a family of expanders the spectral gap goes to 0. It is known that if one takes a sequence of random regular graphs, then with very high probability, it will be a family of expanders (in fact with good expansion).

Expanders have many applications in mathematics, computer science and engineering. Detailed surveys can be found in the *Bulletin of the AMS*, "Expander graphs and their applications" by Hoory, Linial and Wigderson (2006), or the 2003 course notes by Linial and Wigderson of the same name (available online), which have a strong computer science flavor. Applications to circuit design, error-correcting codes and analysis of randomized algorithms are discussed. There is also a more recent 2012 *Bulletin* article by Lubotzky, "Expander graphs in pure and applied mathematics" with more of a bent towards pure mathematics (e.g., number theory and group theory). These surveys are all quite long and technical, and require more mathematical sophistication than we have assumed in these notes (sorry, these are the main introductory references I'm familiar with, and the subject nature itself is quite technical), but may be worth a glance if you're interested in getting a sense of some non-obvious applications, and want more meat than the Wikipedia entry.

Exercises

Exercise 3.4.1. *Let P_n be the path graph on $V = \{1, 2, \dots, n\}$. Consider a random walk on P_n starting at vertex 1.*

(i) Let p_t denote the probability that we reach vertex n for the first time at time t. Compute this (for all t) for $n = 3, 4, 5$.

(ii) For $n = 3, 4, 5$, compute the expected amount of time it will take to reach vertex n— this is given by $\sum_{t=1}^{\infty} p_t t$. (If you can't evaluate this sum exactly, use a computer to estimate it numerically.)

(iii) How do you think the expected amount of time to reach vertex n grows with n (e.g., linearly, quadratically)?

Exercise 3.4.2. *Verify that K_n is a Ramanujan graph.*