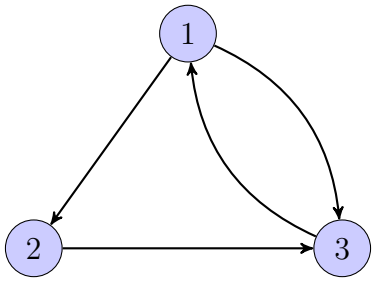
Directed Graphs and The Google Matrix

Consider a directed graph where the vertices represent pages on the internet. Directed edges in the graph represent hyperlinks from one page to another. That is, if there a link from page 1 to page 2 then there is a directed edge in the graph from v_1 to v_2.

Consider the following directed graph



The adjacency matrix for a directed graph has a 1 in the (i, j)-position if there is a directed edge that starts at v_i and ends at v_j:

$$A_{ij} = \begin{cases} 1 & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

Example 8: The adjacency matrix for the graph above is $A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$

For directed graphs we can talk about the *incoming degree* or *outgoing degree* of a vertex. The incoming degree of a vertex is the number of edges that terminate at the vertex. Similarly the outgoing degree of a vertex is the number of edges that start at the vertex. For our purposes, we'll only care about the outgoing degree.

The outgoing degrees of the three vertices in the example graph are given by

$$d_1 = 2 \qquad d_2 = 1 \qquad d_3 = 1$$

Note that the outgoing degree of a vertex is equal to the sum of the associated row in the adjacency matrix.

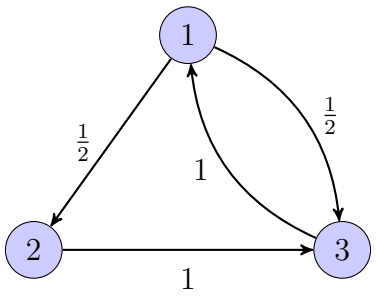
Def: The transition matrix of a directed graph T is found by transposing the adjacency matrix and dividing each column by the outgoing degree of the associated vertex. Mathematically, we have

$$T_{ij} = \frac{A_{ji}}{d_j}$$

For the example graph, we have

$$T = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{bmatrix}$$

The transition matrix gives us an interesting way of interpreting the graph. Consider dropping a *random walker* on a vertex of the graph. The walker then moves along a directed edge to another vertex with a probability evenly distributed between each outgoing edges. For the example graph the probabilities look as follows



Let's see what happens when we drop a random walker at vertex 1 and then let it go.

Step 0: at v_1 with prob $p_1 = 1$

Step 1: at v_2 with prob $p_2 = 1/2$, at v_3 with prob $p_3 = 1/2$

Step 2: at v_1 with prob $p_1 = 1/2$, at v_3 with prob $p_3 = 1/2$

Step 3: at v_1 with prob $p_1 = 1/2$, at v_2 with prob $p_2 = 1/4$, at v_3 with prob $p_3 = 1/4$

The transition matrix T gives us a way to determine the probabilities of the walker using linear algebra. Since we begin with the walker at vertex 1 with probability 1 we denote the state of the system by the vector

$$\mathbf{x}^{(0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

To move the walker one step we multiply the state vector by T. We have

$$\mathbf{x}^{(1)} = T\mathbf{x}^{(0)} = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \end{bmatrix}$$

Note that the state vector after step one corresponds to the probabilities we deduced above for the random walker. To get the state of the system after two steps we multiply again by the transition matrix.

$$\mathbf{x}^{(2)} = T\mathbf{x}^{(1)} = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

which again agrees with the previously computed probabilities. Taking one more step, we have

$$\mathbf{x}^{(3)} = T\mathbf{x}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/4 \\ 1/4 \end{bmatrix}$$

Now suppose that we let the random walker go for a large number of steps until the state vector no longer changes from step to step. The resulting vector is as follows

$$\mathbf{x} = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.4 \end{bmatrix}$$

This says that after a large number of time there is a 40% chance that the walker is at vertex 1, a 20% chance that it's at vertex 2, and a 40% chance that it's at vertex 3.

It's easy to check that the given vector is an eigenvector of the transition matrix T with associated eigenvalue $\lambda = 1$.

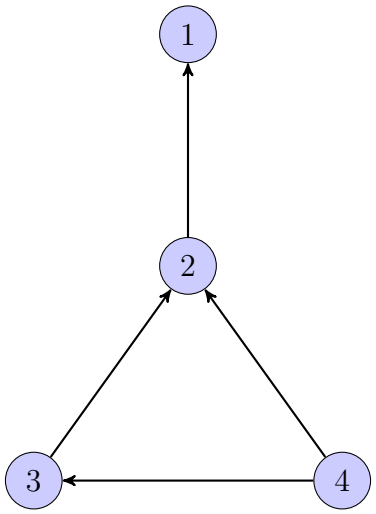
$$\begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0.2 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 + 0.2 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.4 \end{bmatrix}$$

Note that any nonzero scalar multiple of this vector would also be an eigenvector with associated eigenvalue 1. When we interpret the vector as a probability distribution we always scale the vector so that it has all nonnegative entries that sum to 1.

If we return to the analogy that the graph represents a network of webpages, then the eigenvector of interest indicates the long term behavior of a web surfer that is randomly clicking on hyperlinks. The pages that have high probabilities are deemed more important than the others, and would therefore be displayed higher up in the list by Google. The eigenvector is called the **PageRank** vector and the i^{th} entry in the vector is called the PageRank of page i.

Note that this was a simple case where the transition matrix T was a stochastic matrix (all nonnegative entries with column sums of 1). But this isn't always the case for every directed graph.

Consider the following digraph



The associated adjacency matrix is $A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$

The outgoing degrees of the vertices are

$$d_1 = 0 \qquad d_2 = 1 \qquad d_3 = 1 \qquad d_4 = 2$$

Ignoring the zero outgoing degree of vertex 1, the transition matrix T is given by

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/2 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Suppose we start with a random walker at vertex 4. Then the first several state vectors are

$$\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{x}^{(1)} = T\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(2)} = T\mathbf{x}^{(1)} = \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \\ 0 \end{bmatrix}$$

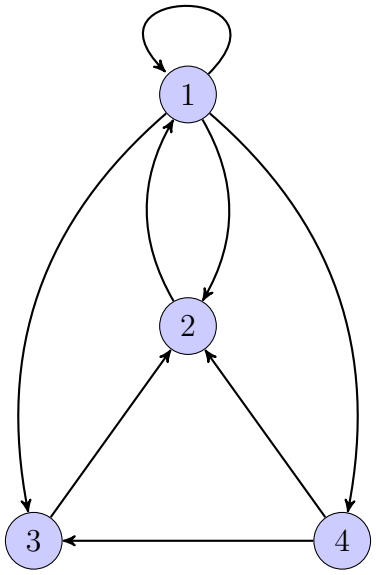
$$\mathbf{x}^{(3)} = T\mathbf{x}^{(2)} = \begin{bmatrix} 1/2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(4)} = T\mathbf{x}^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We immediately see a problem at Step 3. The resulting state vector is no longer a probability distribution because the entries of the vector don't sum to 1.

The problem is that vertex 1 does not have any outgoing nodes, so the probability of the random walker doesn't have anywhere to go and sort of falls off the edge. We call vertex 1 a **dangling vertex** or **dangling node**.

Note also that the transition matrix T for the graph with the dangling vertex is not a stochastic matrix. The column associated with the dangling vertex is all zeros and thus does not sum to 1.

OK, so how do we fix this? Google's solution is to add a directed edge from the dangling vertex to every other vertex in the graph (including itself). The web analogy is that when a random surfer reaches a dangling node, it jumps randomly to any page in the web with equal probability. After fixing the dangling node, the directed graph looks as follows



The associated transition matrix for the new graph is as follows:

$$T = \begin{bmatrix} 1/4 & 1 & 0 & 0 \\ 1/4 & 0 & 1 & 1/2 \\ 1/4 & 0 & 0 & 1/2 \\ 1/4 & 0 & 0 & 0 \end{bmatrix}$$

Notice that the new transition matrix is in fact stochastic. It's PageRank vector is

$$\mathbf{x} = \begin{bmatrix} 0.42 \\ 0.32 \\ 0.15 \\ 0.10 \end{bmatrix}$$

OK, so the PageRank vector is the eigenvector associated with eigenvalue $\lambda = 1$.

- How do we know that a stochastic matrix T always has an eigenvalue $\lambda = 1$?
- How that there isn't an eigenvalue bigger than 1?
- Are there multiple eigenvectors with associated eigenvalue $\lambda = 1$?

Fact 5: If T is column-stochastic then $\lambda = 1$ is an eigenvalue.

Before we can prove this we need to prove an important lemma about eigenvalues.

Lemma: If λ is an eigenvalue of A then it is also an eigenvalue of A^T.

Proof: Recall that the determinant of a matrix and its transpose are the same. We then have

$$\det(A - \lambda I) = \det(A - \lambda I)^T = \det\left(A^T - \lambda I\right)$$

This means that A and A^T have the same characteristic polynomial. Since the eigenvalues of a matrix are the roots of its characteristic polynomial it must be the case that the eigenvalues of A and A^T are the same.

Caveat: Even though the eigenvalues of A and A^T are the same it is usually **NOT** the case that their associated eigenvectors are the same.

OK, now we're ready to prove Fact 5.

Proof: Let $\mathbf{1}$ be the vector of all 1's. Since the columns of T sum to 1 we have

$$T^T\mathbf{1} = \mathbf{1} \quad \Rightarrow \quad \lambda = 1 \text{ is an eigenvalue of } T \text{ and } T^T$$

□

Fact 6: If T is column-stochastic then it has no eigenvalue λ such that $|\lambda| > 1$.

Proof: Suppose that λ is an eigenvalue of T and T^T. Let $\mathbf{w}$ be the eigenvector of T^T associated with λ. Then we have $T^T\mathbf{w} = \lambda\mathbf{w}$. Suppose that w_i is the largest entry in $\mathbf{w}$ (in absolute value). From the equation, we have

$$|\lambda w_i| = \left|\sum_j [T^T]_{ij} w_j\right| = \left|\sum_j T_{ji} w_j\right|$$

Note that the sum over j on the right is just the dot product of $\mathbf{w}$ with the i^{th} row of T^T, which is also the i^{th} column of T.

Next, I'm going to pull the λ out of the absolute values on the left. But the result should still be positive, so I better put an absolute value around the λ as well.

$$|\lambda|\,|w_i| = \left|\sum_j T_{ji} w_j\right|$$

Now, we've assumed that w_i is the largest absolute entry in $\mathbf{w}$. Since the term on the right is a dot product of a column of T with $\mathbf{w}$, and we know that T is column-stochastic, it's easy to see that the largest the right-hand side can be is if the i^{th} column of T is the vector that has a 1 in the i^{th} position, and zeros every else. If the i^{th} column of T is any other vector with all nonnegative entries that sums to 1 the expression on the right will be smaller. Thus we have

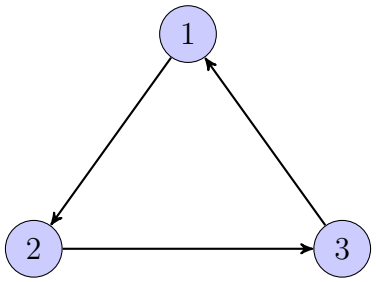
$$|\lambda|\,|w_i| = \left|\sum_j T_{ji} w_j\right| \leq |w_i|$$

Cancelling the $|w_i|$ on both sides of the inequality gives $|\lambda| \leq 1$, which was what we were after.

□

The last question we posed was whether T could have multiple eigenvectors associated with $\lambda = 1$. This is an important question if we are to interpret the dominant eigenvector as the PageRank vector. It would be weird if there were multiple PageRank vectors associated with a web.

It turns out that there are a couple of cases that can lead to non-unique dominant eigenvectors. The first is the case when the graph has multiple disconnected components. We've looked at this case before, so we won't beat it to death again. The new case is the following:



which has transition matrix $T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$.

If we drop a random walker at vertex 1 then we have $\mathbf{x}^{(0)} = \mathbf{e}_1$. Then

$$\mathbf{x}^{(1)} = T\mathbf{x}^{(0)} = \mathbf{e}_2 \qquad \mathbf{x}^{(2)} = T\mathbf{x}^{(1)} = \mathbf{e}_3 \qquad \mathbf{x}^{(3)} = T\mathbf{x}^{(2)} = \mathbf{e}_1$$

It's clear that if we let the walker run for a long time, it will never settle down into a stationary state. It will instead skip between being located at each of the vertices with probability 1. When this happens we say that the graph is *periodic*.

It turns out that the reason this happens is because T has three distinct eigenvalues each with magnitude equal to 1. They are

$$\lambda_1 = 1 \qquad \lambda_{2,3} = 0.5 \pm \frac{\sqrt{3}}{2} i$$

OK, so what do we do to avoid these special cases? Google's solution is to **first fix dangling nodes** to obtain a stochastic matrix T and then add outgoing links from each page to every other page in the web. This can be justified intuitively by the notion that occasionally web surfers opt to type a new URL into the browser rather than clicking on a hyperlink. We need to make this modification to the transition matrix in such a way that we still obtained a column-stochastic matrix. We define the new matrix

$$G = (1-p)\,T + p\frac{1}{n}\mathbf{1}\mathbf{1}^T$$

where $\mathbf{1}$ is n-dimensional vector of all 1's. Notice that G is a column-stochastic matrix as well. The parameter p is a tuning parameter that sets the probability that the surfer makes a random jump. The tuning parameter p is usually chosen to be $p = 0.15$.

Computing the PageRank Vector in Practice

OK, so how to we actually compute the PageRank vector (i.e. the eigenvector of G with eigenvalue $\lambda = 1$)? For very small matrices in class we would row-reduce $G - \lambda I$ to find its nullspace vector. In practice this is really not a good idea. Remember that Gaussian-Elimination is an $\mathcal{O}(n^3)$ method, and if G is a many billions $\times$ many billions matrix, this will pretty much take until the end of time. Instead we'll compute it using the random walker analogy we talked about in class. We'll start with a random probability vector as an initial guess, and then apply the matrix G repeatedly to let the random walker move about the graph. When the entries of the vector (i.e. the probabilities of the locations of the walker) stop changing very much, we'll stop and take that vector as the stationary distribution (i.e. the PageRank vector).

This process is actually a very common (although naive) method for finding the eigenvector of a matrix corresponding to it's largest eigenvector. It's called the **Power Method**, and here's why it works:

Let A be a $n \times n$ column-stochastic matrix assume that it has $\lambda_1 = 1$ and all of its other eigenvalues are less than 1.

$$\lambda_1 = 1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

Let $\mathbf{x}^{(0)}$ be a random vector in $\mathbb{R}^n$. Since the eigenvectors of A form a basis, we can write

$$\mathbf{x}^{(0)} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3 + \cdots + \alpha_n\mathbf{v}_n$$

for some set of coefficients $\{\alpha_i\}$. Now define $\mathbf{x}^{(1)}$ as follows

$$\begin{aligned}\mathbf{x}^{(1)} = A\mathbf{x}^{(0)} &= \alpha_1 A\mathbf{v}_1 + \alpha_2 A\mathbf{v}_2 + \alpha_3 A\mathbf{v}_3 + \cdots + \alpha_n A\mathbf{v}_n \\ &= \alpha_1(\lambda_1\mathbf{v}_1) + \alpha_2(\lambda_2\mathbf{v}_2) + \alpha_3(\lambda_3\mathbf{v}_3) + \cdots + \alpha_n(\lambda_n\mathbf{v}_n) \\ &= \alpha_1\mathbf{v}_1 + \alpha_2(\lambda_2\mathbf{v}_2) + \alpha_3(\lambda_3\mathbf{v}_3) + \cdots + \alpha_n(\lambda_n\mathbf{v}_n)\end{aligned}$$

Similarly

$$\begin{aligned}\mathbf{x}^{(2)} = A\mathbf{x}^{(1)} &= A(\alpha_1\mathbf{v}_1 + \alpha_2\lambda_2\mathbf{v}_2 + \alpha_3\lambda_3\mathbf{v}_3 + \cdots + \alpha_n\lambda_n\mathbf{v}_n) \\ &= \alpha_1 A\mathbf{v}_1 + \alpha_2\lambda_2 A\mathbf{v}_2 + \alpha_3\lambda_3 A\mathbf{v}_3 + \cdots + \alpha_n\lambda_n A\mathbf{v}_n \\ &= \alpha_1\mathbf{v}_1 + \alpha_2\lambda_2^2\mathbf{v}_2 + \alpha_3\lambda_3^2\mathbf{v}_3 + \cdots + \alpha_n\lambda_n^2\mathbf{v}_n\end{aligned}$$

Note that every time we hit the new vector by A again, each eigenvector component gets multiplied again by its eigenvalue. After m iterations of the method, we have

$$\mathbf{x}^{(m)} = A\mathbf{x}^{(m-1)} = \alpha_1\mathbf{v}_1 + \alpha_2\lambda_2^m\mathbf{v}_2 + \alpha_3\lambda_3^m\mathbf{v}_3 + \cdots + \alpha_n\lambda_n^m\mathbf{v}_n$$

Remember that we're hoping that the iterate $\mathbf{x}^{(m)}$ is getting closer and closer to the largest eigenvector, $\mathbf{v}_1$. From the above expression it looks like all of the other eigenvectors are hanging around, but you have to think about what happens when you take powers of the eigenvalues. Since all but the first eigenvalue are smaller than one, powers of the eigenvalues get really small as the iteration goes on. Eventually, each power of the form λ_j^m is so small that their corresponding eigenvectors are not expressed in $\mathbf{x}^{(m)}$, and $\mathbf{x}^{(m)}$ is a good approximation of the PageRank vector.

Now, you want to be a little careful here. Even if you started your iteration with a random nonnegative vector that summed to one, roundoff error will probably make it so the iterate is not quite a probability vector anymore. As such, its usually a good idea to re-scale the vector to make it a probability vector. You can either do this after each iteration of the Power Method, or at the very end.

So why is it called the Power Method? It turns out that by repeatedly multiplying iterates by A, we're implicitly taking powers of the matrix. Notice that

$$\mathbf{x}^{(m)} = A\mathbf{x}^{(m-1)} = A^2\mathbf{x}^{(m-2)} = \cdots = A^m\mathbf{x}^{(0)}$$

So the vector after m iterations is the same (in exact arithmetic) vector you would get if you multiplied the starting vector by the m^{th} power of A. Of course, we would never do this in practice, because multiplying matrices is together is tremendously more expensive than multiplying matrices by vectors.

OK, so how do we know when we're done? Remember that we're after the stationary distribution of the random walker. So one good *stopping criterion* is to measure the change between successive iterates and stop when the change is small. One common way to do this would be to, on the m^{th} iteration, look at the norm of the difference between $\mathbf{x}^{(m)}$ and $\mathbf{x}^{(m-1)}$. If $\|\mathbf{x}^{(m)} - \mathbf{x}^{(m-1)}\|$ is less than some small tolerance (I asked you to use 10^{-5} in your homework) then you say the method has converged, and return $\mathbf{x}^{(m)}$ as the approximation of the PageRank vector.

The Power Method with the Actual Google Matrix

OK, one more trick that we should be aware of when working with the actual Google matrix:

$$G = (1-p)\,T + p\frac{1}{n}\mathbf{1}\mathbf{1}^T$$

Notice that the matrix G is really a combination of the original transition matrix T that describes the connections of the web, and the second matrix $\mathbf{1}\mathbf{1}^T/n$ which describes the random teleportations. The teleportation matrix is a *dense* matrix with every entry equal to $1/n$. On the other hand the matrix T is pretty *sparse* because its only nonzero entries correspond to edges between connected vertices. For the real internet, each webpage is on average connected to only ten other pages. This means that a typical column of T will have ten nonzero entries, and many billions of zeros. Clever matrix storage schemes will actually only store the nonzero entries in the matrix, since the zeros won't do anything when the matrix multiplies a vector. This means that for a, say, billion $\times$ billion matrix, we only have to store ten billion nonzero entries. If we were to store all of the zeros too, that would require storing $(10^9)^2 = 10^{18}$ entries (I looked it up, that's a *quintillion*). Now, if we were to actually add the teleportation matrix to T we'd have to store all 10^{18} of those numbers. This seems like a bad idea. Let's see how we can multiply G by a probability vector, without actually forming G.

Let $\mathbf{x}$ be a vector of nonnegative entries that sum to 1. Then

$$G\mathbf{x} = (1-p)\,T\mathbf{x} + \frac{p}{n}\mathbf{1}\mathbf{1}^T\mathbf{x}$$

Notice in the second term we have $\mathbf{1}^T\mathbf{x}$. This is a dot product that acts to sum up the entries of $\mathbf{x}$, which we know to be 1. This gives us

$$G\mathbf{x} = (1-p)\,T\mathbf{x} + \frac{p}{n}\mathbf{1}$$

This expression says that to apply the action of G to a probability vector $\mathbf{x}$, we can multiply $\mathbf{x}$ by $(1-p)\,T$ and then add p/n to each entry in the resultant vector. This allows us to multply vectors by G without ever forming G! I've asked you to use this trick in your homework when implementing your PageRank function.