Addendum to Chapter 2: Graphs as Matrices

Michele Coscia (mcos@itu.dk)

August 13, 2018

Abstract

In this addendum we discuss various useful properties of the adjacency matrix – and its derivatives – when analyzed with linear algebra instruments.

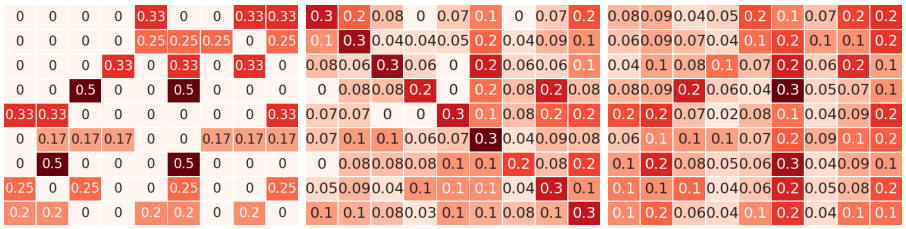
1 Stochastic Adjacency

The adjacency matrix by itself is useful only for a limited set of operations. We usually transform it to squeeze out all the possible analytic juice. The simplest transformation we can perform is to transform it into a stochastic matrix. This means that we normalize it, dividing each entry by the sum of its corresponding row. So if nodes u and v are connected, and u has 5 connections, the A_{uv} entry will be $1/5 = 0.2$. Figure 1 shows an example of this stochastic transformation.

What's the usefulness of the stochastic adjacency matrix? The first direct use we can make of it is to calculate transition probabilities. This is literally what it contains: each entry is the probability that a random walker on a given node (row) will cross that edge. In Figure 1a we see that node 9 has degree equal to five. This corresponds to having five entries set to one in Figure 1b. If we close our eyes and pick one of these at random, each one has a probability of 0.2 to be picked. That is the value in Figure 1c.

(a) (b) (c)

Figure 1: (a) The original graph. (b) The adjacency matrix of (a). (c) The corresponding stochastic version.



(a) A^1 (b) A^2 (c) A^3

Figure 2: Different powers of the stochastic adjacency matrix of the graph in Figure 1a.

Suppose we picked node 6 and that we repeat the exercise. Picking one of node 6's neighbors at random has a probability of 0.17, and we end up – for instance – on node 3. We might want to know what was the likelihood of ending in node 3 starting from node 9 and doing exactly two random jumps. The probability is not simply the product of the two jumps – as we would do naively for independent events – because there is an alternative route. We could have visited node 8 first and *then* moved to 3. We have to keep track of all possible alternative paths, and this becomes really unwieldy when we start considering longer random walks.

Luckily, we don't have to do it. The stochastic matrix has the *power* of telling us what we want. It's literally its *power*. Say A is our stochastic matrix. We just saw how A is just the probability of transitioning from one node to another. In other words, it gives us the probability of all transitions for random walks of length 1. Let's now write this matrix as A^1, which is the same thing as A. Let's say this again: A^1 is the probability of all transitions for random walks of length 1. Could it be, then, that A^2 is the probability of all transitions for random walks of length 2? And that A^n is the probability of all transitions for random walks of length n? Yes, they are! Let's take a closer look at A^2 and A^3 in Figures 2b and 2c.

First, they are both still stochastic matrices, meaning that their rows still sum to 1. So each entry in the matrix is still a transition probability. This time, though, it's not the transition probability of the direct connection, but of a path of length 2 and 3, respectively.

Second, the diagonal is not zero any more. That is because, with a random walk of length 2, there is a chance to select the same edge twice, and therefore returning to the point of origin. So the A^2_{vv} entry tells you the likelihood of starting from v and returning back to v in two steps.

Third, while A^2 still has zero entries, A^3 does not. This is because some node pairs are farther than two edges away, so the probability of a random walker to reach them in two hops is zero – check Figure 1a again if you don't believe me! On the other hand, the diameter of the graph is three, and thus there is always a path of length three between any node pair, no matter how unlikely.

Finally, these matrices are not symmetric any more, while the non-normalized

Figure 3: The result when raising the stochastic A to the power of 30.

adjacency matrix of an undirected graph is. This is because the likelihood of ending in u from v isn't necessarily the same as the other way around: if v has better connected neighbors, the random walkers are more likely to be led astray. For instance, the probability to go from 4 to 5 in three random steps is 0.04, while the other way around is 0.02. That is because node 5 has an extra connection, which would lead the walker to be unable to reach 4 in two additional hops.

There's one curious thing if we look at the columns of the A^1 and A^3 matrices. In the first case, the minimum is zero and the maximum can be up to 0.5. But in A^3 the minimum is higher than zero, and the maximum is just 0.3. It seems that the values are somehow converging. So what happens if we take A^{30} or – gasp! – A^∞?

Figure 3 shows the result. We see now that the columns are constant vectors. These numbers have a specific meaning. When we calculate A^∞, what we're doing is basically asking the probability of ending in a node after a random walk of infinite length. Since the length is infinite, it does not really matter from which node you originally started. That's why all rows of A^∞ are the same – remember that the row indicates the starting point while the column indicates the ending point.

This row vector – you can pick any of them, since they're all the same – is so important that we give it a name. We call it the "stationary distribution" – or π, for short. π tells us that, if you have a path of infinite length, the probability of ending up on a destination is only dependent on the destination's location and not on your point of origin. In practice, if you apply the transition probability (A) to the stationary distribution (π), you still obtain the stationary distribution: $\pi A = \pi$.

This is practically the same as the PageRank, which we described in the previous lecture. Having a high value in the stationary distribution for a node means that you are likely to visit it often with a random walker, which is exactly what PageRank estimates. However, PageRank has to deal with additional problems.

What do you do if your graph is not connected? No matter how many

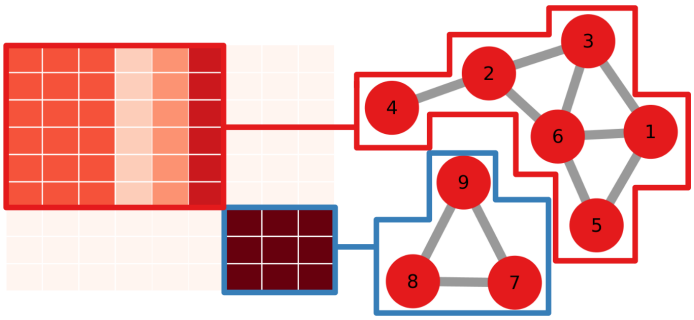


Figure 4: The result when calculating the stationary distribution for an unconnected graph.

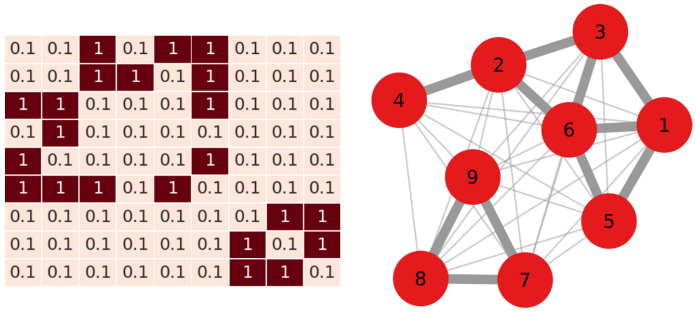


Figure 5: The practical implementation of the PageRank's teleportation trick.

powers of A you take, how infinitely long your walks are, some destinations are unreachable from some origins. We end up with two stationary distributions, one for one component, and one for the other. Figure 4 shows an example.

Google's solution for the PageRank algorithm was to give the walker a teletransportation device. At each step, the walker has a minuscule chance to request a teletransportation, which then might land it on a different component. This mathematical trick is actually embarrassingly easy to implement. It is equivalent to the creation of ghost edges with very little weight connecting the entire graph. On matrix notation, this is the same as adding a tiny constant to the (not yet normalized) adjacency matrix: $A^* = A + \epsilon$. Figure 5 shows this teleportation trick in practice.

A few closing remarks. First, it is not necessary to calculate A^∞ to know the stationary distribution. π is quite literally the normalized degree of the nodes: the degree divided by the sum of all degrees. Second, as a consequence, also PageRank is very close to the degree. How closely the degree approximates the PageRank is dependent on the value of our teleportation parameter ϵ – in the literature, we call $1 - \epsilon$ the "damping factor". The magic value of ϵ is 0.15, that is what Brin and Page used originally. If we set $\epsilon = 0$, PageRank is equivalent to π, and therefore to the degree.

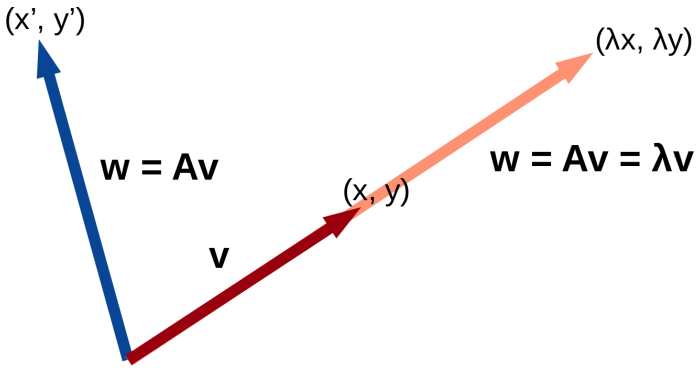


Figure 6: A graphical depiction of an eigenvector.

2 Graph Spectrum

To continue our journey into the fantastic powers of the adjacency matrix we need to dust off some linear algebra. A big caveat here: this is a very superficial recap of only the crucial concepts we need to continue. This is not supposed to be even an introductory section to linear algebra itself.

Let's abridge the concept of eigenvector, based on Figure 6. Given a vector v, we can apply any arbitrary matrix transformation A to it. We then obtain a new vector $w = Av$. Some transformations A are special. If we scale the vector, without altering its direction, it is as if we multiplied its elements by the same scalar λ: $w = \lambda v$. The matrix transformation A that can achieve this exact result is special: v is its eigenvector and λ is its associated eigenvalue. Mathematically, we represent this relation as $Av = \lambda v$.

The formula we just introduced is the one for *right* eigenvectors, because the vector multiplies the matrix *from the right*. As you might expect, there are also *left* eigenvectors, which multiply the matrix *from the left*: $vA = v\lambda$. Right and left eigenvectors are different, have different values, but their corresponding eigenvalues are the same. From now on, when we mention eigenvectors, we refer to the *right* eigenvectors. Right eigenvectors are the *default*, and when we refer to *left* eigenvectors we will explicitly acknowledge it.

Our adjacency matrix A, which has $|V|$ rows and $|V|$ columns, has $|V|$ eigenvalues. Usually, we sort them in decreasing order. The largest eigenvalue of a stochastic adjacency matrix is always equal to one – we also call it the "leading" eigenvalue, or λ_1. The second largest eigenvalue λ_2 could be equal to the first – and therefore to one. This is related to the first application of linear algebra to network analysis. The number of eigenvalues equal to one is the number of the components in the graph. The reason is that you can consider the adjacency matrix as two adjacency matrices pasted into the same. They are disjoint matrices and each has as a maximum eigenvalue one. We show an example in Figure 7.

As we see in Figure 6, each eigenvalue has a corresponding eigenvector.

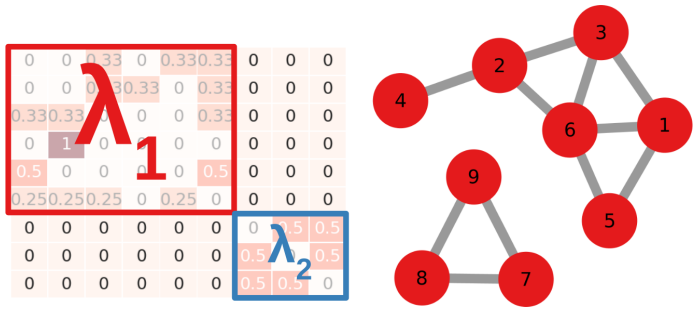


Figure 7: The stochastic adjacency matrix of a disconnected graph looks like two different adjacency matrices pasted on the diagonal. Thus, they both have a (different) leading eigenvalue equal to one.

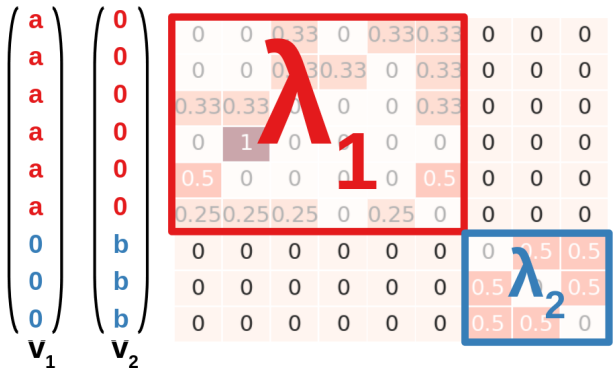


Figure 8: If your graph has two components, the eigenvectors associated with the largest two eigenvalues of the stochastic adjacency matrix will tell you to which component the node belongs, by having a non-zero constant value.

The point of looking at eigenvectors is that there is a relationship between the v-th entry in the i-th eigenvector of an adjecency matrix and node v's relationship with the entire topology of the graph. We're not going into the complex mathematical reasons of how these relationship arise, we are going to look at the most common uses, so that you can understand the terminology. When we sort eigenvalues we can also find the eigenvector associated to the largest eigenvalue – which we call the "largest" or "leading" eigenvector, for convenience.

If you can use the leading eigenvalues to count the number of connected components (Figure 7), the leading eigenvectors tell you to which component the nodes belong. If the network has two components, the nodes belonging to one will have a constant value in the eigenvector, while the nodes which do not belong to that component will have a zero – see Figure 8. As you might have already deduced, if there is only one component then the leading eigenvector

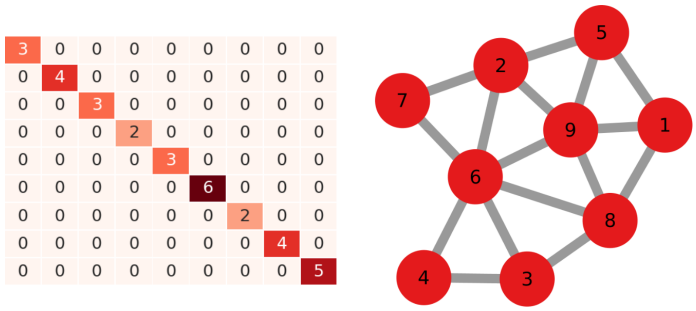


Figure 9: The degree matrix of the sample graph.

contains the same a constant value.

But... wait! We already seen a constant vector somewhere in this section. It was the column vector of the A^{∞}, the transformation we made to the adjacency matrix to find the stationary distribution. The two are indeed connected, but we need to do a little detour to really appreciate their relatedness.

You remember that the definition of a stationary distribution we arrived at was $\pi A = \pi$. This kind of looks like our eigenvector specification ($Av = \lambda v$), with a few odd parts. First, where's the eigenvalue? Well, we can always multiply a vector to 1 and we won't change anything in the equation. So: $\pi A = \pi 1$. This is cool, because we already know that 1 is the largest eigenvalue (λ_1). Second, the vector π is *on the left*, not *on the right*. Putting these things together: the stationary distribution π is the vector associated with the largest eigenvalue, if multiplied on the left of A. Therefore: π is the leading left eigenvector.

Here's the final kicker. If the matrix is symmetric – as any adjacency matrix of an undirected graph is – there is a relationship between right and left eigenvectors. If you were to transpose the adjacency matrix, that is making it column-normalized instead of row-normalized, the left and right eigenvectors would swap. In different words: the left eigenvectors of A are exactly the same as the right eigenvectors of A^T. Thus the vector of constant and π are the right and left leading eigenvectors of A, and they swap roles in A^T.

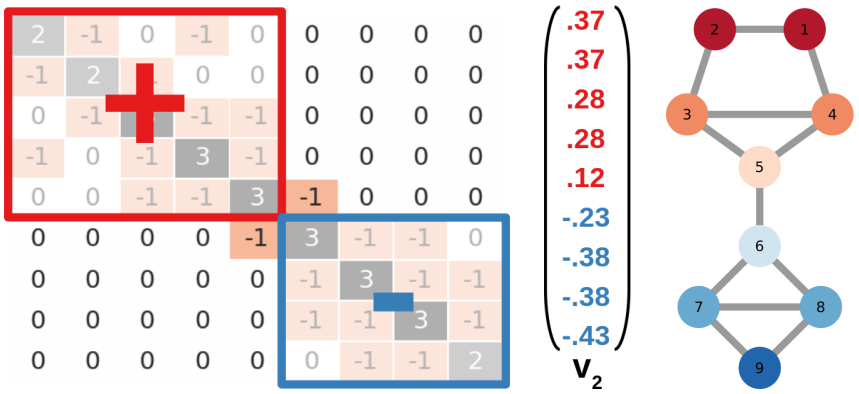
3 Graph Laplacian

If that seems esoteric, wait until you see what you can do with a Graph Laplacian. To know what that is, we need to introduce the concept of Degree Matrix. Usually we call it D. D is a very simple animal. It is a matrix whose diagonal entries are the degrees of the corresponding nodes. The other off-diagonal entries in the matrix are equal to zero. Figure 9 shows an example of a degree matrix.

The Laplacian version of the adjacency matrix – which we call L – is the result of a simple operation: $L = D - A$. In practice, we take the degree matrix D and we subtract A from it. L is a matrix that has the node degree

(a) D (b) A (c) L

Figure 10: The operation producing the Laplacian matrix L (c), subtracting the adjacency matrix A (b) from the degree matrix D (a).

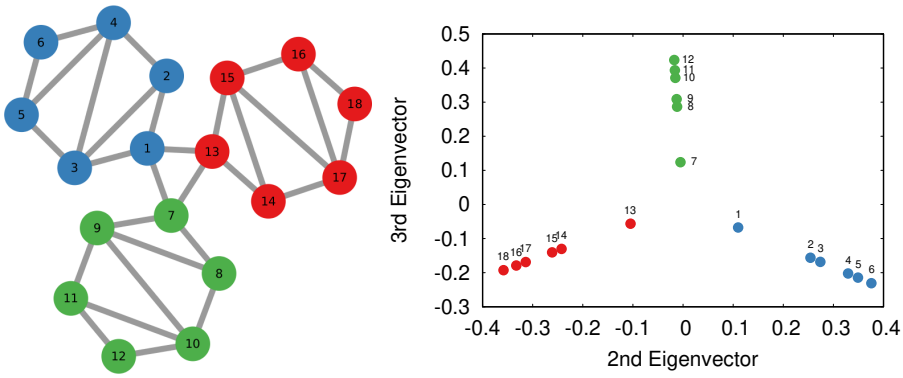


(a) (b) (c)

Figure 11: (a) The Laplacian matrix of a graph. We show the 2-cut solution for this graph with the red and blue blocks. (b) The second smallest eigenvector of (a). (c) The graph view of (a) – we color the nodes according to the value attached to them in the (b) vector.

in the diagonal, -1 for each entry corresponding to an edge in the network, and zero everywhere else. Figure 10 depicts the operation. L has some obvious properties: since it has the degree of the node on the diagonal and -1 for each of the node's connection, the sums of all rows and columns are equal to zero.

Just like for A, we are also interested in the eigenvectors of L. We need to make some adjustments, though. Instead of sorting them in decreasing order, we sort them in increasing order. That is, now λ_1 is the *smallest* eigenvalue, not the largest. It also takes a special value like for A but, for L, $\lambda_1 = 0$. The smallest eigenvector of L plays the very same role as did the largest eigenvector of A: it is a vector of constant, telling us to which component the node belongs. The reason why L is interesting lies with the second smallest eigenvector (or the eigenvector associated to the smallest non-zero eigenvalue, if the graph has multiple components).



(a) (b)

Figure 12: (a) A graph in which the node colors represent the best solution to the 3-cut problem. (b) The eigenspace of the Laplacian of the (a) graph. We plot the second smallest eigenvector in the x-axis, and the third smallest eigenvector on the y-axis. Data point color corresponds to the node color in (a). Each data point is labeled with its corresponding node ID.

One classical problem in graph theory is to find the N-cut of a graph: how to divide nodes in N disjoint groups such that the number of edges running across groups is minimized. Turns out that the second smallest eigenvector of the Laplacian is a very good approximation to solve this problem. How? Consider Figure 11. In Figure 11a we show the Laplacian matrix of a graph. We arranged the rows and columns of the matrix so that the 2-cut solution is evident: by dividing the matrix in two diagonal blocks there is only one edge outside our block structure that needs to be cut.

Now, why did we label the two blocks as "+" and "-"? The reason lies in the magical second smallest eigenvector of the Laplacian, which is in Figure 11b. We can see that the top entries are all positive (in red) and the bottom are all negative (in blue). This is where L shines: by looking at the sign of the value of a node in its second smallest eigenvector we know in which group the node has to be to solve the 2-cut problem!

Not only that, but the values in Figure 11b are clearly in descending order. If we show the graph itself – in Figure 11c – and we use these values as node colors, we discover that there is much more information than that in the eigenvector. The absolute value tells us how embedded the node is in the group, or how far from the cut it is. Node 5 is right next to it, while node 9 is the farthest away.

Now – you're thinking – *you're itching to tell me you can use this eigenvector to solve all the N-cut problems. But you can't, because it's just a simple monodimensional vector.* Or can I? True: the second smallest eigenvector cannot solve, *by itself*, the N-cut problem. That's why we have *all the other eigenvectors* of the Laplacian.

Solving the 3-cut problem involves looking at the eigenvectors in a two di-

mensional space. We show and example of this in Figure 12. Figure 12b is a 2D representation of the second and third smallest eigenvectors of the Laplacian of the graph in Figure 12a. We can see that there is a clear pattern: each node takes a position in this space on a different axis, depending on the block to which it belongs. Farther nodes on the axis are more embedded in the block, while nodes closer to the cuts are nearby the origin (0,0). You can imagine that we could solve the 4-cut problem looking at a 3D space, and the N-cut problem looking at a $(N-1)$D space, but I can't show it because I ran out of Möbius paper.

And now you're thinking with matrices.