Transition matrices

Transition matrices are part of a large area of mathematics (stochastic modeling) which we will not go into detail of. We can use a few basic results to create a Page-Rank algorithm ourselves. You need to understand these reults in order to apply the algebra and interpret your results correctly.

A transition matrix M is an $n \times n$ matrix with elements in the range $[0, 1]$, such that all columns sum to 1. You can think of the element in $M[i, j]$ representing the probability of moving from state j to state i over one time step. Transition matrices can model systems where internet users are moving between different websites, the weather changes form day to day, consumers switch between brands, citizens moving between income brackets, and many more applications in applied math, physics, biology, engineering, sociology, etc...

A distribution vector $\vec{x}$ is a vector with elements in the range $[0, 1]$ which sum to 1. We can think of a distribution vector as a current known or estimated distribution. We can then use our transition matrix to compute the expected distribution after one timestep: $M\vec{x_t} = \vec{x_{t+1}}$. If you are interested, you can show that if $\vec{x_t}$ is a distribution vector, then so is $\vec{x_{t+1}}$.

To make our lives easy, we add the restriction that M is positive, meaning that all of its entries are > 0. This guarantees that the matrix behaves as needed. We need the following result:

If M is a positive[1] transition matrix, then there is a unique equilibrium distribution vector $\vec{x_{eq}}$, which has the two properties:

I $M\vec{x_{eq}} = \vec{x_{eq}}$.

II $M^t\vec{x_0} \xrightarrow{t\to\infty} \vec{x_{eq}}$, for any initial distribution vector $\vec{x_0}$.

The mathematical notation may look intimidating, but the intuition is straightforward. No matter how we distribute, for example, internet users over the web, the distribution eventually settles into a stable distribution. In the stable distribution, the individual users are moving around from node to node, but the proportion of users at each page is stable across transitions. The stable distribution is an excellent approximation of how popular webpages are be relative to each other.

The Page-Rank Algorithm (simplified)

As stated in the previous section, we can use the equilibrium distribution of users on the internet as an approximation for the relative popularity of web pages. We start with an adjacency matrix, N, of which the elements represent links between pages. If a page has a number of links to other pages, we assume that a user picks one of them uniformly at random and moves to the next page. However, internet users do not navigate the web only by following links. (If you end up on a page with no links on it, you do not get stuck there for the rest of your life.) Users also occasionally move between pages with no links on them. We call this teleportation, and to simulate it in our model, we simply add a small constant p to every element of N, and call the new matrix N_p. This will translate to their being a non-zero probability of moving between any pair of pages, whether or not there is a link. This is not only a better model with respect to reality, it is also computationally beneficial,

[1] This statement is true for all so-called regular transition matrices, which are matrices where M^k is positive for some constant k, but this is a harder condition to check, and our application will use positive matrices anyway.

since we now have a positive matrix. When we normalize the columns of N_p (scale them so that the sum is 1), we have a positive transition matrix N_t, which means that we can find the equilibrium vector $\vec{x_{eq}}$. This is our vector of Page-Rank scores!

In the previous section, we saw that there are two ways of finding $\vec{x_{eq}}$, because of the two properties I and II. The first one asks for an eigenvector with eigenvalue 1, which we can then scale to ensure it is a distribution vector. (You may want to look up the definition of eigenvectors and eigenvalues, but you do not need to know much of this.) The properties of positive transition matrices ensure that any eigenvector with eigenvalue 1 will be a scaling of the equilibrium vector. We can ask numpy for the set of eigenvalues and eigenvectors of a matrix, by using the command

```
from numpy import linalg as LA
eigvals,eigvecs=LA.eig.(Nt)
```

This command gives the `eigvals` as a list, and the `eigvecs` as a matrix, of which the columns are the eigenvectors. The indices of the column correspond to the indices of the eigenvalues in the list. So, if we wish to find the eigenvector with eigenvalue 1, we need to find the index of the 1 in the list of `eigvals`, and then use that index to retrieve the right column from `eigvecs`.

Numpy does not order the eigenvalues, but we know (again, from the properties of positive transition matrices), that the 1 is the unique largest (in terms of absolute values) eigenvalue in the list. This is helpful, because numpy does numerical approximations, which means that we cannot look for the exact value 1, which may have been stored as .9995 or 1.0002. So, to find the index of the 1 in the list of eigenvalues, we ask

```
abseigvals=abs(eigvals)
ind=abseigvals.index(max(abseigvals))
```

Once we have this index, we find the corresponding column in eigvecs, scale it so that the sum is 1, and we have $\vec{x_{eq}}$! The elements of this vector represent the popularity of pages, by giving the probability that a user is on a page at any given time. From here, we can then find the ranking vector, which lists the most popular page, the second most popular page, the third... Exactly like the result of a Google search.