USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

Las matemáticas del algoritmo PageRank

Ángela Piñeiro Lourés

2019/2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

Las matemáticas del algoritmo PageRank

Ángela Piñeiro Lourés

Febrero, 2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área de Coñecemento: Álgebra.
Título: Las matemáticas del algoritmo PageRank.
Breve descrición do contido
El algoritmo PageRank es la base del método usado por Google para dar respuesta a una consulta que se está haciendo en la web. El objetivo de este trabajo es explicar las matemáticas que subyacen en este algoritmo. Con este fin se estudiarán conceptos algebraicos asociados a un grafo, el método de la potencia, la teoría de Perron-Frobenius para matrices no negativas y los conceptos elementales de la teoría de cadenas de Markov.
Recomendacións
Outras observacións

Índice general

Resumen

En 1998, Larry Page y Sergey Brin revolucionaron la eficiencia de los motores de búsqueda web con la creación del algoritmo PageRank, todavía hoy usado por el motor de búsqueda de Google. A principios de los años 90, los motores de búsqueda usaban sistemas de clasificación basados en texto para decidir qué páginas eran más relevantes para una consulta dada. La idea que el algoritmo PageRank sacó a colación fue que la importancia de cualquier página web puede ser juzgada observando el número y la autoridad de las páginas que enlazan con ella. Traduciendo estas palabras en términos matemáticos entra en juego el álgebra lineal, mientras que la teoría de grafos y las cadenas de Markov están en la esencia del algoritmo.

Abstract

In 1998, Larry Page and Sergey Brin revolutionized the efficiency of web search engines by creating the PageRank algorithm, still used by Google's search engine today. In the early 90s, the search engines used text-based ranking systems to decide which pages are most relevant to a given query. The idea that PageRank brought up was that the importance of any web page could be judged by looking at the number and authority of the pages that link to it. Translating these words into mathematical terms brings linear algebra into play, while graph theory and Markov chains are at the heart of the algorithm.

Introducción

Vivimos en una era informática. Internet forma parte de nuestra vida cotidiana y la información está a sólo un clic de distancia. Basta abrir nuestro motor de búsqueda web favorito, escribir las palabras clave, y el motor de búsqueda nos mostrará las páginas relevantes para nuestra consulta. ¿Pero cómo funciona realmente un motor de búsqueda?

A primera vista, parece razonable imaginar que un motor de búsqueda tiene un índice de todas las páginas web y cuando un usuario escribe una consulta, el motor navega en su índice y cuenta las apariciones de las palabras clave en cada fichero web. Las ganadoras son las páginas con el número más alto de incidencias de las palabras clave.

Este solía ser el escenario correcto a principios de los años 90, cuando los motores de búsqueda web usaban sistemas de clasificación basados en texto para decidir qué páginas eran más relevantes. Sin embargo, estos sistemas eran muy ineficaces puesto que devolvían miles de páginas web que el usuario debía revisar para determinar cuáles eran de su interés. Por ejemplo, si realizábamos una búsqueda del término "internet" la primera página que nos mostraba uno de los primeros motores estaba escrita en chino, con repetidas incidencias de la palabra internet y ninguna otra información.

La utilidad de un motor de búsqueda depende de la relevancia del conjunto de resultados que devuelve. Puede que haya millones de páginas web que contengan una palabra concreta o frase, sin embargo algunas de ellas serán más relevantes, populares o autorizadas que otras. Uno espera que las páginas relevantes se muestren dentro del top 20 de las páginas devueltas.

Los motores de búsqueda modernos emplean métodos de clasificación más elaborados que la ordenación basada en texto y devuelven al usuario una lista ordenada donde las páginas más relevantes se encuentran en cabeza. Uno de los algoritmos más conocidos para calcular la relevancia de las páginas web es el algoritmo PageRank, usado por el motor de búsqueda de Google. Para una consulta dada el algoritmo calcula un vector, denominado vector PageRank, que contiene las puntuaciones de importancia para cada página web. Fue inventado por Larry Page y Sergey Brin, dos doctorandos de la Universidad de Stanford, y se convirtió en una marca registrada por Google en 1998. La idea que planteó el PageRank

es que la importancia de una página web puede ser juzgada observando las páginas web que enlazan con ella. Si creamos una página web i e incluimos un hiperenlace a la página web j, esto significa que consideramos j importante para nuestro tema. Si hay muchos enlaces a la página j, entonces la creencia común es que la página j es relevante. Si por el contrario j tiene sólo un enlace entrante pero viene de una página prestigiosa y autorizada k, decimos que k transfiere autoridad a j. Ya sea que hablemos de popularidad o autoridad, podemos asignar un rango a cada página web, basándonos en los rangos de las páginas que enlazan con ella.

El contenido de este trabajo se divide en cuatro capítulos estructurados de la siguiente forma.
Empezaremos el capítulo 1 tratando los conceptos básicos de álgebra lineal que surgen en el análisis matemático del algoritmo PageRank. Estudiaremos los autovalores y autovectores de una matriz y explicaremos su forma de Jordan. A continuación, introduciremos dos modos útiles y equivalentes de interpretar las funciones de matrices. Una de ellas, conocida como "teorema espectral para matrices", será de utilidad para comprender el método de la potencia, que trataremos en la sección siguiente. Este método iterativo fue el método elegido por Google para calcular el vector PageRank. En la tercera sección desarrollaremos la teoría de Perron-Frobenius, definiendo conceptos asociados como matriz irreducible y matriz primitiva. Esta teoría nos permitirá asegurar la existencia de un único vector PageRank positivo. Y gracias a la primitividad, obtendremos además la convergencia del método de la potencia aplicado al problema PageRank. Para terminar el capítulo, y dado que nuestro vector PageRank resultará ser el vector probabilidad estacionario de una cadena de Markov, nos familiarizaremos con los conceptos asociados a la teoría de cadenas de Markov. Esta teoría nos permitirá ajustar el algoritmo para garantizar la convergencia.

En el capítulo 2 haremos un breve recorrido por la historia de la búsqueda de información, que fue revolucionada en 1989 por el nacimiento de la World Wide Web, una particular colección de documentos repleta de enlaces. Comentaremos las dificultades iniciales que encontraban los usuarios cuando intentaban realizar una consulta en la Web. Y cómo la aparición en 1998 del análisis de enlaces mejoró drásticamente los resultados ofrecidos por los motores de búsqueda web. En la segunda sección, señalaremos las características particulares de la Web con respecto a las colecciones tradicionales de documentos. Para terminar el capítulo 2, describiremos brevemente los elementos que participan en el sistema de búsqueda web con el fin de contextualizar el proceso de clasificación de las páginas web.

Comenzaremos el capítulo 3 contextualizando el algoritmo PageRank y explicando la idea intuitiva subyacente. A continuación, plantearemos cuál fue la fórmula original y qué motivos llevaron a sus creadores a optar por un proceso iterativo. En la siguiente sección, desarrollaremos una fórmula matricial iterativa más compacta y a continuación, comentaremos los diversos problemas de convergencia que surgen entorno al proceso iterativo. Para terminar, analizaremos cuáles fueron los ajustes realizados por Brin y Page para asegurar resultados de convergencia deseables y descubriremos por qué el método de la potencia fue el método elegido para calcular el vector PageRank.

En el capítulo 4 incluiremos algunos ejemplos de cálculo del vector PageRank realizados en SageMath.

Capítulo 1

Preliminares

1.1. Álgebra lineal

1.1.1. Autovalores y autovectores

Para una matriz $A \in \mathbb{C}^{n\times n}$, los escalares λ y los vectores $x_{n\times 1} \neq 0$ que verifican $Ax = \lambda x$ son respectivamente los *autovalores* y los *autovectores* de A. Un vector fila y^T es un *autovector por la izquierda* si $y^T A = \lambda y^T$.

El conjunto $\sigma(A)$ de los distintos autovalores es llamado *espectro* de A, y el *radio espectral* de A es el número no negativo

$$\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda|.$$

La circunferencia del plano complejo centrada en el origen y con radio $\rho(A)$ es llamada *circunferencia espectral*, y es sencillo verificar que

$$\rho(A) \leq \|A\|$$

para cualquier norma matricial.

Los autovalores de $A_{n\times n}$ son las raíces del polinomio característico $p(\lambda) = \det(A - \lambda I)$, donde $\det(\star)$ denota el determinante. El grado de $p(\lambda)$ es n, así que A tiene n autovalores en total, pudiendo haber autovalores complejos (incluso si las entradas de la matriz A son números reales) y también autovalores repetidos. Si A contiene sólo números reales, entonces sus autovalores complejos deben aparecer como pares conjugados, i.e. si $\lambda \in \sigma(A)$, entonces $\bar{\lambda} \in \sigma(A)$.

La *multiplicidad algebraica* de un autovalor λ de A es el número de veces que λ se repite como raíz de la ecuación característica. Si $\text{alg mult}_A(\lambda) = 1$, entonces λ es un *autovalor simple*.

La *multiplicidad geométrica* de un autovalor λ de A es el número de autovectores linealmente independientes que están asociados con λ. En términos formales, geo $\text{mult}_A(\lambda) = \dim \ker(A - \lambda I)$, donde $\ker(\star)$ denota el núcleo de una matriz. Es siempre cierto que geo $\text{mult}_A(\lambda) \leq$ alg $\text{mult}_A(\lambda)$. Si geo $\text{mult}_A(\lambda) =$ alg $\text{mult}_A(\lambda)$, entonces se dice que λ es un *autovalor semisimple*.

El *índice de un autovalor* de la matriz A, $\lambda \in \sigma(A)$, es el entero positivo k más pequeño tal que $\text{rank}((A - \lambda I)^k) = \text{rank}((A - \lambda I)^{k+1})$. Se sobrentiende que $\text{index}(\lambda) = 0$ cuando $\lambda \notin \sigma(A)$.

Hay varias formas de caracterizar el índice. Para $\lambda \in \sigma(A_{n\times n})$, decir que $k = \text{index}(\lambda)$ es equivalente a decir que k es el entero positivo más pequeño tal que cualquiera de las siguientes afirmaciones se verifica.

1. $\text{rank}((A - \lambda I)^k) = \text{rank}((A - \lambda I)^{k+1})$.
2. $\ker((A - \lambda I)^k) = \ker((A - \lambda I)^{k+1})$.
3. $\text{rank}((A - \lambda I)^k) \cap \ker((A - \lambda I)^k) = 0$.
4. $\mathbb{C}^n = \text{rank}((A - \lambda I)^k) \oplus \ker((A - \lambda I)^k)$.

1.1.2. La forma de Jordan

La forma de Jordan será una herramienta útil para tratar los conceptos subyacentes a la búsqueda web puesto que nos permite caracterizar completamente los autovalores y autovectores de una matriz cuadrada A.

Dada una matriz $A_{n\times n}$, un *bloque de Jordan* asociado a un autovalor $\lambda \in \sigma(A)$ se define como una matriz de la forma

$$J_\star(\lambda) = \begin{pmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{pmatrix}. \tag{1.1}$$

Un *segmento de Jordan* $J(\lambda)$ asociado a $\lambda \in \sigma(A)$ es una matriz diagonal por bloques conteniendo uno o más bloques de Jordan. Formalmente, un segmento de Jordan es

$$J(\lambda) = \begin{pmatrix} J_1(\lambda) & 0 & \cdots & 0 \\ 0 & J_2(\lambda) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_t(\lambda) \end{pmatrix} \tag{1.2}$$

siendo cada $J_\star(\lambda)$ un bloque de Jordan.

La *forma canónica de Jordan* (o simplemente *forma de Jordan*) para A es una matriz diagonal por bloques compuesta por los segmentos de Jordan para los distintos autovalores. En otras palabras, si $\sigma(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_s\}$, entonces la forma de Jordan de A es

$$J = \begin{pmatrix} J(\lambda_1) & 0 & \cdots & 0 \\ 0 & J(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J(\lambda_s) \end{pmatrix}. \tag{1.3}$$

El siguiente teorema nos permite conocer el tamaño y el número de los bloques de Jordan (véase [3, p. 590]).

Teorema 1.1 (Teorema de Jordan)**.** *Para cada $A \in \mathbb{C}^{n\times n}$ existe una matriz no singular P tal que*

$$P^{-1}AP = J \tag{1.4}$$

es la forma de Jordan (1.3) *y está caracterizada por las siguientes propiedades.*

1. *J contiene un segmento de Jordan $J(\lambda)$ por cada autovalor distinto $\lambda \in \sigma(A)$.*
2. *Cada segmento $J(\lambda)$ contiene $t = \dim\ker(A - \lambda I)$ bloques de Jordan.*
3. *El número de bloques de Jordan $i \times i$ está dado por*

$$\nu_i(\lambda) = r_{i-1}(\lambda) - 2r_i(\lambda) + r_{i+1}(\lambda), \quad \textit{donde} \quad r_i(\lambda) = \operatorname{rank}((A - \lambda I)^i).$$

4. *El bloque de Jordan más grande en cada segmento $J(\lambda)$ es $k \times k$, donde $k = \operatorname{index}(\lambda)$.*

La estructura de J es única en el sentido que el número y el tamaño de los bloques de Jordan en cada segmento está únicamente determinado por las entradas de A. Dos matrices A y B de dimensión $n \times n$ son *semejantes* (i.e. $B = Q^{-1}AQ$ para alguna matriz Q no singular) si y sólo si A y B tienen la misma forma de Jordan.

La matriz P en (1.4) no es única, pero sus columnas siempre forman *cadenas de Jordan* (o *autovectores generalizados*) en el siguiente sentido. Para cada bloque de Jordan $J_\star(\lambda)$ hay un conjunto de columnas $P_\star$ con su correspondiente tamaño y posición en $P = (\cdots | P_\star | \cdots)$ tal que

$$P_\star = ((A - \lambda I)^i x_\star \mid (A - \lambda I)^{i-1} x_\star \mid \cdots \mid (A - \lambda I)x_\star \mid x_\star)_{(i+1)\times n}$$

para algún i y algún $x_\star$, donde $(A - \lambda I)^i x_\star$ es un autovector particular asociado a λ. Existen fórmulas para calcular i y $x_\star$ [3, p. 594], pero los cálculos pueden ser complicados. Afortunadamente, rara vez necesitaremos calcular P.

Observación 1.2. Recordemos que la multiplicidad algebraica es el número de veces que λ se repite como raíz de la ecuación característica y la multiplicidad geométrica es el número de autovectores linealmente independientes asociados a λ, es decir, geo $\text{mult}_A(\lambda) = \dim \ker(A - \lambda I)$. Entonces, es importante observar que podemos usar la forma de Jordan para encontrar directamente las multiplicidades algebraica y geométrica.

Para un autovalor λ, la multiplicidad algebraica es igual a la suma de los tamaños de todos los bloques de Jordan en $J(\lambda)$. La multiplicidad geométrica es igual al número de bloques de Jordan asociados con λ. Cuando $\text{index}(\lambda) = 1$ tenemos que el bloque de Jordan más grande es de tamaño 1×1. En este caso, de (1.2) se sigue que alg $\text{mult}_A(\lambda) =$ geo $\text{mult}_A(\lambda)$, es decir, λ es un autovalor semisimple.

Veamos un importante corolario del teorema de Jordan 1.1 relativo a la diagonalización de una matriz cuadrada.

Corolario 1.3. *Cada una de las siguientes afirmaciones es equivalente a decir que $A \in \mathbb{C}^{n\times n}$ es semejante a una matriz diagonal, i.e. J es diagonal (todos sus bloques de Jordan son 1×1).*

1. $\text{index}(\lambda) = 1$ *para cada* $\lambda \in \sigma(A)$.

2. alg $\text{mult}_A(\lambda) =$ geo $\text{mult}_A(\lambda)$ *para cada* $\lambda \in \sigma(A)$.

3. *A tiene un conjunto completo de n autovectores linealmente independientes (i.e. cada columna de P es un autovector de A).*

1.1.3. Funciones de una matriz

Un importante uso de la forma de Jordan es definir funciones de $A \in \mathbb{C}^{n\times n}$. Es decir, dada una función $f\colon \mathbb{C} \longrightarrow \mathbb{C}$, ¿qué significado tiene $f(A)$? La respuesta es bastante simple. Supongamos que $A = PJP^{-1}$, donde $J = \begin{pmatrix} \ddots & & \\ & J_\star & \\ & & \ddots \end{pmatrix}$ está en forma de Jordan con los $J_\star$ representando los bloques de Jordan descritos en (1.1). Entonces es natural definir el valor de f en A como

$$f(A) = Pf(J)P^{-1} = P \begin{pmatrix} \ddots & & \\ & f(J_\star) & \\ & & \ddots \end{pmatrix} P^{-1}, \tag{1.5}$$

pero para ello es imprescindible definir $f(J_\star)$ correctamente. Resulta que la definición adecuada es

$$f(J_\star) = f\begin{pmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{pmatrix} = \begin{pmatrix} f(\lambda) & f'(\lambda) & \frac{f''(\lambda)}{2!} & \cdots & \frac{f^{(k-1)}(\lambda)}{(k-1)!} \\ & f(\lambda) & f'(\lambda) & \ddots & \vdots \\ & & \ddots & \ddots & \frac{f''(\lambda)}{2!} \\ & & & f(\lambda) & f'(\lambda) \\ & & & & f(\lambda) \end{pmatrix}. \tag{1.6}$$

Definición 1.4. Sea $A \in \mathbb{C}^{n\times n}$ con $\sigma(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_s\}$, y sea $f\colon \mathbb{C} \longrightarrow \mathbb{C}$ tal que $f(\lambda_i), f'(\lambda_i), \ldots, f^{(k_i-1)}(\lambda_i)$ existen para cada i, donde $k_i = \text{index}(\lambda_i)$. Definimos

$$f(A) = Pf(J)P^{-1} = P\begin{pmatrix} \ddots & & \\ & f(J_\star) & \\ & & \ddots \end{pmatrix} P^{-1}, \tag{1.7}$$

donde J es la forma de Jordan para A y $f(J_\star)$ está dada por (1.6).

Existe otra manera útil y equivalente de ver las funciones de matrices. Se conoce como *teorema espectral para funciones de matrices* (véase [3, p. 603]) y surge como resultado del desarrollo del producto en el lado derecho de la ecuación (1.7). Antes de introducirla consideremos la siguiente definición.

Definición 1.5. Una *proyección o proyector* sobre un espacio vectorial V es una aplicación lineal $P\colon V \longrightarrow V$ tal que $P^2 = P$. Si se tiene una proyección sobre el espacio vectorial V y se denota por N el núcleo de P y por W el rango de P se verifican las siguientes propiedades.

1. P es la aplicación identidad Id restringida a W, es decir, $P(x) = x$ para todo $x \in W$.
2. Tenemos una suma directa de espacios vectoriales $V = W \oplus N$.

Se dice que P es una *proyección sobre W a lo largo de N*.

Teorema 1.6 (Teorema espectral para matrices generales)**.** *Si $A \in \mathbb{C}^{n\times n}$ con $\sigma(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_s\}$, entonces*

$$f(A) = \sum_{i=1}^{s}\sum_{j=0}^{k_i-1} \frac{f^{(j)}(\lambda_i)}{j!}(A - \lambda_i I)^j G_i, \tag{1.8}$$

donde cada G_i tiene las siguientes propiedades.

- G_i *es un proyector (i.e.* $G_i^2 = G_i$*) sobre* $\ker((A - \lambda_i I)^{k_i})$ *a lo largo de* $\operatorname{rank}((A - \lambda_i I)^{k_i})$.
- $G_1 + G_2 + \cdots + G_s = I$.
- $G_i G_j = 0$ *cuando* $i \neq j$.
- $(A - \lambda_i I)G_i = G_i(A - \lambda_i I)$ *es nilpotente con índice* k_i.

Los G_i *son llamados proyectores espectrales asociados a la matriz* A.

Si A es diagonalizable, i.e. si es semejante a una matriz diagonal

$$A = P \begin{pmatrix} \lambda_1 I & 0 & \cdots & 0 \\ 0 & \lambda_2 I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_s I \end{pmatrix} P^{-1},$$

entonces

$$f(A) = P \begin{pmatrix} f(\lambda_1) I & 0 & \cdots & 0 \\ 0 & f(\lambda_2) I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\lambda_s) I \end{pmatrix} P^{-1},$$

y la fórmula (1.8) da lugar al siguiente teorema espectral para matrices diagonalizables (véase [3, p. 517]).

Teorema 1.7 (Teorema espectral para matrices diagonalizables)**.** *Si* A *es diagonalizable con* $\sigma(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_s\}$*, entonces*

$$A = \lambda_1 G_1 + \lambda_2 G_2 + \cdots + \lambda_s G_s, \tag{1.9}$$

y

$$f(A) = f(\lambda_1) G_1 + f(\lambda_2) G_2 + \cdots + f(\lambda_s) G_s, \tag{1.10}$$

donde los proyectores espectrales G_i *tienen las siguientes propiedades.*

- $G_i = G_i^2$ *es el proyector sobre el autoespacio* $\ker(A - \lambda_i I)$ *a lo largo de* $\operatorname{rank}(A - \lambda_i I)$.
- $G_1 + G_2 + \cdots + G_s = I$.
- $G_i G_j = 0$ *cuando* $i \neq j$.
- $G_i = \prod_{\substack{j=1 \\ j \neq i}}^{k} (A - \lambda_j I) / \prod_{\substack{j=1 \\ j \neq i}}^{k} (\lambda_i - \lambda_j)$ *para* $i = 1, 2, \ldots, k$.

- *Si λ_i es un autovalor simple, entonces*

$$G_i = xy^T/y^Tx \tag{1.11}$$

donde x e y^T son respectivamente los autovectores por la derecha y por la izquierda asociados a λ_i.

1.2. El método de la potencia

El método elegido por Google para calcular el vector PageRank fue el *método de la potencia*. Se trata de una técnica iterativa que calcula el autopar dominante (λ_1, x) de una matriz $A \in \mathbb{R}^{m\times m}$ diagonalizable con autovalores

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_k|.$$

Observamos que la hipótesis $|\lambda_1| > |\lambda_2|$ implica que λ_1 es real, sino $\bar{\lambda_1}$ (el complejo conjugado) es otro autovalor con el mismo módulo que λ_1. Consideramos la función $f(z) = (z/\lambda_1)^n$ y usando la representación espectral (1.10) junto con la condición $|\lambda_i/\lambda_1| < 1$ para $i = 2, 3, \ldots, k$ concluimos que

$$\begin{aligned}\left(\frac{A}{\lambda_1}\right)^n &= f(A) = f(\lambda_1)G_1 + f(\lambda_2)G_2 + \cdots + f(\lambda_k)G_k \\ &= G_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^n G_2 + \cdots + \left(\frac{\lambda_k}{\lambda_1}\right)^n G_k \to G_1 \text{ cuando } n \to \infty. \end{aligned} \tag{1.12}$$

Para cada x_0 tenemos $(A^nx_0/\lambda_1^n) \to G_1x_0 \in \ker(A - \lambda_1 I)$, así que, si $G_1x_0 \neq 0$, entonces A^nx_0/λ_1^n converge a un autovector asociado a λ_1. Esto significa que la dirección de A^nx_0 tiende hacia la dirección de un autovector, dado que λ_1^n actúa sólo como factor de escala para mantener la longitud de A^nx_0 bajo control. En lugar de utilizar λ_1^n, podemos escalar A^nx_0 de formas más convenientes. Por ejemplo, $\|A^nx_0\|$ (para cualquier norma vectorial) es un factor de escala razonable, pero existen mejores opciones. Para un vector v, sea $m(v)$ la componente de magnitud máxima, y si hay más de una componente maximal, sea $m(v)$ la primera. Por ejemplo, $m(1, 3, -2) = 3$, y $m(-3, 3, -2) = -3$. Es claro que $m(\alpha v) = \alpha m(v)$ para cualquier escalar α. Supongamos que $m(A^nx_0/\lambda_1^n) \to \gamma$. Dado que $(A^n/\lambda_1^n) \to G_1$, se sigue que

$$\lim_{n\to\infty} \frac{A^nx_0}{m(A^nx_0)} = \lim_{n\to\infty} \frac{(A^n/\lambda_1^n)x_0}{m(A^nx_0/\lambda_1^n)} = \frac{G_1x_0}{\gamma} = x$$

es un autovector asociado con λ_1. Pero en vez de calcular progresivamente potencias de A, la sucesión $A^nx_0/m(A^nx_0)$ se genera en modo más eficiente tomando $x_0 \notin \text{rank}(A - \lambda_1 I)$

y definiendo

$$y_n = Ax_n, \quad \nu_n = m(y_n), \quad x_{n+1} = \frac{y_n}{\nu_n}, \quad para\ n = 0, 1, 2, \ldots \tag{1.13}$$

No sólo se verifica que $x_n \to x$, sino que además obtenemos que $\nu_n \to \lambda_1$ porque para todo n, $Ax_{n+1} = A^2 x_n/\nu_n$, así que si $\nu_n \to \nu$ cuando $n \to \infty$, el límite en el lado izquierdo es $Ax = \lambda_1 x$, mientras que el límite en lado derecho es $A^2 x/\nu = \lambda_1^2 x/\nu$. Como estos dos límites deben coincidir, $\lambda_1 x = (\lambda_1^2/\nu)x$, y esto implica que $\nu = \lambda_1$. El método de la potencia puede ser sintetizado como sigue.

Teorema 1.8. *Sea x_0 arbitrario. (En realidad, no puede ser completamente arbitrario dado que necesitamos que $x_0 \notin \operatorname{rank}(A - \lambda_1 I)$ para asegurar que $G_1 x_0 \neq 0$, pero es altamente improbable que eligiendo un vector arbitrario x_0 resulte $G_1 x_0 = 0$). Si definimos*

$$y_n = Ax_n, \quad \nu_n = m(y_n), \quad x_{n+1} = \frac{y_n}{\nu_n}, \quad para\ n = 0, 1, 2, \ldots, \tag{1.14}$$

entonces $x_n \to x$ y $\nu_n \to \lambda_1$, donde $Ax = \lambda_1 x$.

Adelantamos algunas de las razones por las que el método de la potencia resulta atractivo para calcular el vector PageRank de Google.

1. Cada iteración requiere sólo de un producto matriz-vector, y esto puede ser aprovechado para reducir el esfuerzo computacional cuando A es grande y dispersa (la mayor parte de sus elementos son cero), como será nuestro caso.

2. Los cálculos pueden ser hechos en paralelo mediante el cómputo simultáneo de productos internos de filas de A con x_n.

3. Resulta claro de (1.12) que, para una matriz diagonalizable, el ritmo al que (1.14) converge depende de cómo de rápido $(\lambda_2/\lambda_1)^n \to 0$. Veremos que Google puede regular $|\lambda_2|$ y por tanto, controlar el ritmo de convergencia.

4. Adelantamos que $\lambda_1 = 1$ para el problema PageRank de Google, por tanto no hay necesidad de utilizar el factor de escala ν_n. En otras palabras, las iteraciones son simplemente $x_{n+1} = Ax_n$.

Introducimos a continuación la implementación del método de la potencia en SageMath. Tomemos, por ejemplo,

$$A = \begin{pmatrix} -7 & -12 \\ 8 & 13 \end{pmatrix} \quad \text{y} \quad x_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

```
from numpy import argmax,argmin
A=matrix([[-7,-12],[8,13]])
x0=vector([1.0,0.0]) # autovector inicial
maxit=20 # número máximo de iteraciones
dig=8 # número de cifras mostradas
tol=0.0001       # límite de tolerancia para la diferencia de \
   dos autovectores consecutivos
err=1 # inicialización de la tolerancia
i=0
while(i<=maxit and err>=tol):
        y0=A*x0
        ymod=y0.apply_map(abs)
        imax=argmax(ymod)
        c1=y0[imax]
        x1=y0/c1
        err=norm(x0-x1)
        i=i+1
        x0=x1
        print("Número de iteración:", i-1)
        print("y"+str(i-1)+"=",y0.n(digits=dig),
        "c"+str(i-1)+"=",c1.n(digits=dig),
        "x"+str(i)+"=",x0.n(digits=dig))
        print
```

```
Número de iteración: 0
y0= (-7.0000000, 8.0000000), c0= 8.0000000, x1= (-0.87500000, 1.0000000)

Número de iteración: 1
y1= (-5.8750000, 6.0000000), c1= 6.0000000, x2= (-0.97916667, 1.0000000)

Número de iteración: 2
y2= (-5.1458333, 5.1666667), c2= 5.1666667, x3= (-0.99596774, 1.0000000)

Número de iteración: 3
y3= (-5.0282258, 5.0322581), c3= 5.0322581, x4= (-0.99919872, 1.0000000)

Número de iteración: 4
```

```
y4= (-5.0056090, 5.0064103), c4= 5.0064103, x5= (-0.99983995, 1.0000000)

Número de iteración: 5
y5= (-5.0011204, 5.0012804), c5= 5.0012804, x6= (-0.99996800, 1.0000000)

Número de iteración: 6
y6= (-5.0002240, 5.0002560), c6= 5.0002560, x7= (-0.99999360, 1.0000000)
```

De este resultado, observamos que el autovalor dominante es 5 y el autovector correspondiente al autovalor dominante es $(-1, 1)$.

1.3. La teoría de Perron-Frobenius

Se trata de una de las teorías más importantes que se encuentran en el mundo de las matemáticas debido tanto a su elegancia como a su amplia utilidad.

Definición 1.9. Una matriz $A \in \mathbb{R}^{n\times n}$ se dice *no negativa* cuando todas sus entradas son números no negativos (se denota como $A \geq 0$). Análogamente, una matriz A se dice una *matriz positiva* cuando cada $a_{ij} > 0$ (se escribe $A > 0$).

La teoría de Perron-Frobenius revela las propiedades de estos dos tipos de matrices describiendo la naturaleza de sus autovalores y autovectores dominantes. Dado que las matrices que forman los cimientos del algoritmo PageRank serán de estos dos tipos, esta teoría se sitúa en el foco de nuestro interés.

1.3.1. Perron

Esta sección recoge el teorema de Perron de 1907 que proporciona un análisis de la estructura espectral para matrices positivas. Más tarde, Georg Frobenius (1912) demostró la extensión de parte de este teorema a ciertas clases de matrices no negativas.

Para empezar, veamos unas observaciones elementales que nos serán de utilidad más adelante. Primero, observamos que

$$A > 0 \quad \Longrightarrow \quad \rho(A) > 0 \tag{1.15}$$

porque si $\sigma(A) = \{0\}$, entonces la forma de Jordan de A, y por tanto A, son nilpotentes, lo cual es imposible cuando cada $a_{ij} > 0$. Esto significa que nuestra discusión puede ser limitada a matrices positivas con radio espectral igual a 1 porque A siempre puede ser

normalizada por su radio espectral, i.e. $A > 0 \iff A/\rho(A) > 0$, y $\rho(A) = r \iff \rho(A/r) = 1$. Otras observaciones fácilmente verificables son

$$P > 0,\ x \geq 0,\ x \neq 0 \quad \Longrightarrow \quad Px > 0, \tag{1.16}$$

$$N \geq 0,\ u \geq v \geq 0 \quad \Longrightarrow \quad Nu \geq Nv, \tag{1.17}$$

$$N \geq 0,\ z > 0,\ Nz = 0 \quad \Longrightarrow \quad N = 0, \tag{1.18}$$

$$N \geq 0,\ N \neq 0,\ u > v > 0 \quad \Longrightarrow \quad Nu > Nv. \tag{1.19}$$

En lo sucesivo, la notación $|\star|$ será usada para denotar una matriz de valores absolutos, i.e. $|M|$ es la matriz con entradas $|m_{ij}|$. Por último, notar que como consecuencia de la desigualdad triangular, es siempre cierto que $|Ax| \leq |A|\ |x|$.

Teorema 1.10. *Si $A_{n\times n} > 0$, entonces las siguientes afirmaciones son ciertas.*

- $\rho(A) \in \sigma(A)$.
- *Si $Ax = \rho(A)x$, entonces $A|x| = \rho(A)|x|$ y $|x| > 0$.*

En otras palabras, A tiene un autopar de la forma $(\rho(A), v)$ con $v > 0$.

Demostración. Como mencionamos anteriormente, podemos asumir sin pérdida de generalidad que $\rho(A) = 1$. Si (λ, x) es cualquier autopar para A tal que $|\lambda| = 1$, entonces

$$|x| = |\lambda|\ |x| = |\lambda x| = |Ax| \leq |A|\ |x| = A\ |x| \quad \Longrightarrow \quad |x| \leq A\ |x|. \tag{1.20}$$

El objetivo es mostrar que se verifica la igualdad. Por comodidad, sea $z = A\ |x|$ e $y = z - |x|$, y observamos que la desigualdad (1.20) implica $y \geq 0$. Supongamos que $y \neq 0$, es decir, existe algún $y_i > 0$. En este caso, de la observación (1.16) se sigue que $Ay > 0$ y $z > 0$, así que debe existir un número $\epsilon > 0$ tal que $Ay > \epsilon z$ o, equivalentemente,

$$\frac{A}{1+\epsilon} z > z.$$

Escribiendo la desigualdad como $Bz > z$, donde $B = A/(1+\epsilon)$, y multiplicando sucesivamente ambos lados por B mientras usamos (1.19) obtenemos

$$B^2 z > Bz > z, \quad B^3 z > B^2 z > z, \quad \ldots \quad \Longrightarrow \quad B^k z > z \quad \text{para todo } k = 1, 2, \ldots$$

Pero $\lim_{k\to\infty} B^k = 0$ porque $\rho(B) = \rho(A/(1+\epsilon)) = 1/(1+\epsilon) < 1$ (véase [3, p. 617, (7.10.5)]), así que en el límite $0 > z$, que contradice el hecho que $z > 0$. Dado que la suposición $y \neq 0$ nos conduce a una contradicción, la suposición debe ser falsa y, en consecuencia, $0 = y = A|x| - |x|$. Por tanto, $|x|$ es un autovector para A asociado al autovalor $1 = \rho(A)$. La prueba se completa observando que $|x| = A|x| = z > 0$. □

Ahora que hemos establecido que $\rho(A) > 0$ es un autovalor para $A > 0$, el siguiente paso es determinar el índice de este autovalor especial.

Teorema 1.11. *Si $A_{n\times n} > 0$, entonces las siguientes afirmaciones son ciertas.*

- *$\rho(A)$ es el único autovalor de A en la circunferencia espectral.*
- *$index(\rho(A)) = 1$. En otras palabras, $\rho(A)$ es un autovalor semisimple. Véase la observación 1.2 (p. 4).*

Demostración. De nuevo, asumimos sin pérdida de generalidad que $\rho(A) = 1$. Sabemos por la segunda afirmación del teorema 1.10 que si (λ, x) es un autopar para A tal que $|\lambda| = 1$, entonces $0 < |x| = A\,|x|$, así que $0 < |x_k| = (A\,|x|)_k = \sum_{j=1}^{n} a_{kj}|x_j|$. Por otro lado, es también cierto que $|x_k| = |\lambda|\,|x_k| = |(\lambda x)_k| = |(Ax)_k| = |\sum_{j=1}^{n} a_{kj}x_j|$, y por tanto

$$\Big|\sum_j a_{kj}x_j\Big| = \sum_j a_{kj}|x_j| = \sum_j |a_{kj}x_j|. \tag{1.21}$$

Para vectores no nulos $\{z_1, \ldots, z_n\} \subset \mathbb{C}^n$, se verifica que $\|\sum_j z_j\|_2 = \sum_j \|z_j\|_2$ (desigualdad triangular) si y sólo si cada $z_j = \alpha_j z_1$ para algún $\alpha_j > 0$ (véase [3, p. 277, Exercise 5.1.10]). En particular, esto es cierto para escalares, así que la ecuación (1.21) asegura la existencia de números $\alpha_j > 0$ tales que

$$a_{kj}x_j = \alpha_j(a_{k1}x_1) \quad \text{o, equivalentemente,} \quad x_j = \pi_j x_1 \quad \text{con } \pi_j = \frac{\alpha_j a_{k1}}{a_{kj}} > 0.$$

En otras palabras, si $|\lambda| = 1$, entonces $x = x_1 p$, donde $p = (1, \pi_2, \ldots, \pi_n)^T > 0$, así que

$$\lambda x = Ax \quad \Longrightarrow \quad \lambda p = Ap = |Ap| = |\lambda p| = |\lambda| p = p \quad \Longrightarrow \quad \lambda = 1,$$

y por tanto 1 es el único autovalor de A en la circunferencia espectral. Ahora supongamos que index$(1) = m > 1$. Se sigue que $\|A^k\|_\infty \longrightarrow \infty$ cuando $k \longrightarrow \infty$ porque hay un bloque de Jordan $J_\star$ de dimensión $m \times m$ en la forma de Jordan $J = P^{-1}AP$ tal que

$$J_\star^k = \begin{pmatrix} 1 & \binom{k}{1} & \cdots & \binom{k}{m-1} \\ & \ddots & \ddots & \vdots \\ & & \ddots & \binom{k}{1} \\ & & & 1 \end{pmatrix}.$$

Por tanto, $\|J_\star^k\|_\infty \longrightarrow \infty$, que a su vez significa que $\|J^k\|_\infty \longrightarrow \infty$, y en consecuencia, $\|J^k\|_\infty = \|P^{-1}A^kP\|_\infty \leq \|P^{-1}\|_\infty \|A^k\|_\infty \|P\|_\infty$ implica

$$\|A^k\|_\infty \geq \frac{\|J^k\|_\infty}{\|P^{-1}\|_\infty \|P\|_\infty} \longrightarrow \infty.$$

Sea $A^k = (a_{ij}^{(k)})$, y sea i_k el índice de la fila para la cual $\|A^k\|_\infty = \sum_j a_{i_k j}^{(k)}$. Sabemos que existe un vector $p > 0$ tal que $p = Ap$, así que para tal autovector,

$$\|p\|_\infty \geq p_{i_k} = \sum_j a_{i_k j}^{(k)} p_j \geq \left(\sum_j a_{i_k j}^{(k)} \right) (\min_i p_i) = \|A^k\|_\infty (\min_i p_i) \longrightarrow \infty.$$

Pero esto es imposible porque p es un vector constante, así que la suposición index$(1) > 1$ tiene que ser falsa, y por tanto index$(1) = 1$. □

Establecer que $\rho(A)$ es un autovalor semisimple de $A > 0$ fue sólo un pequeño escalón, aunque importante, para llegar al siguiente teorema relativo a las multiplicidades de $\rho(A)$.

Teorema 1.12. *Si $A_{n\times n} > 0$, entonces* $\text{alg mult}_A(\rho(A)) = 1$. *En otras palabras, el radio espectral de A es un autovalor simple de A. Por tanto,* $\dim \ker(A - \rho(A)I) = \text{geo mult}_A(\rho(A)) = \text{alg mult}_A(\rho(A)) = 1$.

Demostración. Como antes, asumimos sin pérdida de generalidad que $\rho(A) = 1$, y supongamos que $\text{alg mult}_A(\lambda = 1) = m > 1$. Ya sabemos que $\lambda = 1$ es un autovalor semisimple, lo que significa que $\text{alg mult}_A(1) = \text{geo mult}_A(1)$ (véase p. 2), por tanto hay m autovectores linealmente independientes asociados con $\lambda = 1$. Si x e y son un par de autovectores independientes asociados con $\lambda = 1$, entonces $x \neq \alpha y$ para cualquier $\alpha \in \mathbb{C}$. Seleccionamos una componente no nula de y, sea $y_i \neq 0$, y definimos $z = x - (x_i/y_i)y$. Como $Az = z$, sabemos por la segunda afirmación del teorema 1.10 que $A|z| = |z| > 0$. Pero esto contradice que $z_i = x_i - (x_i/y_i)y_i = 0$. Por tanto, la suposición que $m > 1$ debe ser falsa, y en conclusión $m = 1$. □

Como $\ker(A - \rho(A)I)$ es un espacio de dimensión uno que puede ser generado por algún $v > 0$, hay un *único* autovector $p \in \ker(A - \rho(A)I)$ tal que $p > 0$ y $\sum_j p_j = 1$ (es obtenido por la normalización $p = v/\|v\|_1$). Este autovector especial p es llamado *vector de Perron* para $A > 0$, y el autovalor asociado $r = \rho(A)$ es llamado *raíz de Perron* de A.

Como $A > 0 \iff A^T > 0$, y dado que $\rho(A) = \rho(A^T)$, es cierto que si $A > 0$, entonces además del autopar de Perron (r, p) para A existe un autopar de Perron correspondiente (r, q) para A^T. Puesto que se verifica $q^T A = rq^T$, el vector $q^T > 0$ es llamado *vector de Perron por la izquierda* para A.

Aunque los restantes autovalores de A distintos de $\rho(A)$ pueden ser o no ser positivos, resulta que los únicos autovectores positivos (o incluso no negativos) son los múltiplos positivos del vector de Perron.

Teorema 1.13. *Los únicos autovectores no negativos para $A_{n\times n} > 0$ son el vector de Perron p y sus múltiplos positivos.*

Demostración. Si (λ, y) es un autopar para A tal que $y \geq 0$, y si $x > 0$ es el vector de Perron para A^T, entonces $x^T y > 0$ por (1.16), así que

$$\rho(A)x^T = x^T A \quad \Longrightarrow \quad \rho(A)x^T y = x^T Ay = \lambda x^T y \quad \Longrightarrow \quad \rho(A) = \lambda.$$

□

En 1942, el matemático alemán Lothar Collatz (1910–1990) descubrió la siguiente fórmula para la raíz de Perron, y en 1950 Helmut Wielandt la utilizó para desarrollar la teoría de Perron-Frobenius.

Teorema 1.14 (Fórmula Collatz-Wielandt)**.** *La raíz de Perron de $A_{n\times n} > 0$ viene dada por $r = \max_{x\in\mathcal{N}} f(x)$, donde*

$$f(x) = \min_{\substack{1\leq i\leq n \\ x_i\neq 0}} \frac{(Ax)_i}{x_i} \quad y \quad \mathcal{N} = \{x \mid x \geq 0 \text{ con } x \neq 0\}.$$

donde $(Ax)_i$ representa la i-ésima coordenada de Ax.

Demostración. Si $\xi = f(x)$ para $x \in \mathcal{N}$, entonces $0 \leq \xi x \leq Ax$. Sean p y q^T respectivamente los vectores de Perron de A por la derecha y por la izquierda asociados con la raíz de Perron r y usamos (1.17) con $q^T x > 0$ (por (1.16)) para escribir

$$\xi x \leq Ax \quad \Longrightarrow \quad \xi q^T x \leq q^T Ax = rq^T x \quad \Longrightarrow \quad \xi \leq r \quad \Longrightarrow \quad f(x) \leq r \quad \forall\, x \in \mathcal{N}.$$

Como $f(p) = r$ y $p \in \mathcal{N}$, se sigue que $r = \max_{x\in\mathcal{N}} f(x)$. □

Ejemplo 1.15. La función $f(x)$ está definida en $\mathbb{R}^n$ a excepción del origen. Es homogénea de grado 0 ($f(\alpha x) = f(x)$ para todo $x \in \mathbb{R}^n \backslash \{0\}$).

Consideremos por ejemplo la matriz

$$\begin{pmatrix} 4 & 2 \\ 1 & 5 \end{pmatrix}.$$

La función f está definida en $\mathcal{N} = \{(x_1, x_2) \mid x_1, x_2 \geq 0\} \setminus \{0, 0\}$. La homogeneidad puede ser útil para simplificar los cálculos. En $\mathbb{R}^2$ podemos expresar la función en una única variable considerándola por ejemplo en el segmento $\{(t, 1-t) \mid 0 \leq t \leq 1\}$.

Para $t = 0$ tenemos

$$f(0,1) = \frac{\left(\begin{pmatrix} 4 & 2 \\ 1 & 5 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}\right)_2}{\begin{pmatrix} 0 \\ 1 \end{pmatrix}_2} = \frac{5}{1} = 5.$$

Para $t = 1$ tenemos

$$f(1,0) = \frac{\left(\begin{pmatrix} 4 & 2 \\ 1 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right)_1}{\begin{pmatrix} 1 \\ 0 \end{pmatrix}_1} = \frac{4}{1} = 4.$$

Para $0 < t < 1$ tenemos

$$\begin{aligned} f(t,1-t) &= \text{mín}\left(\frac{\left(\begin{pmatrix} 4 & 2 \\ 1 & 5 \end{pmatrix}\begin{pmatrix} t \\ 1-t \end{pmatrix}\right)_1}{\begin{pmatrix} t \\ 1-t \end{pmatrix}_1}, \frac{\left(\begin{pmatrix} 4 & 2 \\ 1 & 5 \end{pmatrix}\begin{pmatrix} t \\ 1-t \end{pmatrix}\right)_2}{\begin{pmatrix} t \\ 1-t \end{pmatrix}_2}\right) \\ &= \text{mín}\left(2 + \frac{2}{t}, 4 + \frac{1}{1-t}\right). \end{aligned}$$

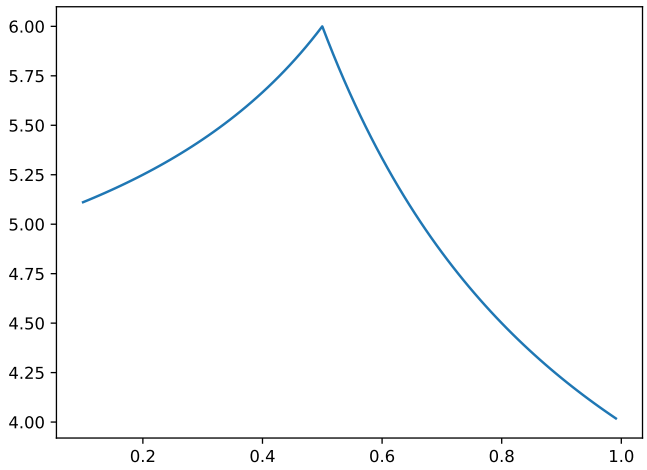


Figura 1.1: La función de Collatz-Wielandt

Comparando las dos funciones racionales en el intervalo $(0,1)$ se sigue que

$$f(t,1-t) = \begin{cases} 4 + \frac{1}{1-t}, & \text{si } 0 < t < \frac{1}{2}, \\ 2 + \frac{2}{t}, & \text{si } \frac{1}{2} < t < 1. \end{cases}$$

En la figura 1.1 podemos ver el gráfico de la función de Collatz-Wielandt para el ejemplo. Observamos que el valor máximo es 6 y se alcanza en $t = \frac{1}{2}$. Un cálculo directo de autovalores muestra que la matriz tiene autovalores reales 3 y 6.

A continuación se incluye el teorema de Perron, que recoge los resultados obtenidos en esta sección.

Teorema 1.16. *Si $A_{n\times n} > 0$ con $r = \rho(A)$, entonces las siguientes afirmaciones son ciertas.*

1. $r > 0$.
2. $r \in \sigma(A)$ *(r es llamada la raíz de Perron).*
3. $\text{alg mult}_A(r) = 1$ *(la raíz de Perron es simple).*
4. *Existe un autovector $x > 0$ tal que $Ax = rx$.*
5. *El vector de Perron es el único vector definido por*

$$Ap = rp, \quad p > 0, \quad y \quad \|p\|_1 = 1,$$

   y, a excepción de los múltiplos positivos de p, no hay otros autovectores no negativos para A, independientemente del autovalor.

6. *r es el único autovalor en la circunferencia espectral de A.*
7. $r = \text{máx}_{x\in\mathcal{N}} f(x)$ *(la fórmula de Collatz-Wielandt),*

$$\textit{donde } f(x) = \min_{\substack{1\le i\le n \\ x_i\neq 0}} \frac{(Ax)_i}{x_i} \quad y \quad \mathcal{N} = \{x \mid x \geq 0 \textit{ con } x \neq 0\}.$$

1.3.2. Extensión a matrices no negativas

Es natural preguntarse qué ocurre si permitimos que A tenga entradas nulas. El siguiente teorema afirma que una parte del teorema de Perron para matrices positivas puede ser extendido a matrices no negativas sacrificando la existencia de un autovector positivo por uno no negativo.

Teorema 1.17. *Para $A_{n\times n} \geq 0$ con $r = \rho(A)$, las siguientes afirmaciones son ciertas.*

1. $r \in \sigma(A)$, *(pero $r = 0$ es posible).*
2. *Existe un autovector $z \geq 0$ tal que $Az = rz$.*
3. *La fórmula de Collatz-Wielandt permanece válida.*

Demostración. Consideramos la sucesión de matrices positivas $A_k = A+(1/k)E > 0$, donde E es la matriz formada enteramente por unos, y sean $r_k > 0$ y $p_k > 0$ respectivamente la raíz de Perron y el vector de Perron para A_k. Observamos que $\{p_k\}_{k=1}^{\infty}$ es un conjunto acotado porque está contenido en la esfera unitaria de $\mathbb{R}^n$. El teorema de Bolzano-Weierstrass establece que toda sucesión acotada en $\mathbb{R}^n$ tiene una subsucesión convergente. Por tanto, $\{p_k\}_{k=1}^{\infty}$ tiene una subsucesión convergente

$$\{p_{k_i}\}_{i=1}^{\infty} \to z, \text{ donde } z \geq 0 \text{ con } z \neq 0 \text{ (porque } p_{k_i} > 0 \text{ y } \|p_{k_i}\|_1 = 1).$$

Como $A_1 > A_2 > \cdots > A$, entonces $r_1 \geq r_2 \geq \cdots \geq r$ (véase [3, p. 619, Example 7.10.2]). Por tanto, $\{r_k\}_{k=1}^{\infty}$ es una sucesión monótona de números positivos acotada inferiormente por r. Un resultado estándar de análisis garantiza que

$$\lim_{k\to\infty} r_k = r^\star \quad \text{existe, y } r^\star \geq r. \quad \text{En particular,} \quad \lim_{i\to\infty} r_{k_i} = r^\star \geq r.$$

Pero $\lim_{k\to\infty} A_k = A$ implica $\lim_{i\to\infty} A_{k_i} = A$, así que usando el hecho de que el límite de un producto es el producto de los límites (siempre que todos los límites existan), es también cierto que

$$Az = \lim_{i\to\infty} A_{k_i}p_{k_i} = \lim_{i\to\infty} r_{k_i}p_{k_i} = r^\star z \quad \Longrightarrow \quad r^\star \in \sigma(A) \quad \Longrightarrow \quad r^\star \leq r.$$

En consecuencia, $r^\star = r$, y $Az = rz$ con $z \geq 0$ y $z \neq 0$. Por tanto, las dos primeras afirmaciones quedan demostradas. Para probar la última, sea $q_k^T > 0$ el vector de Perron por la izquierda de A_k. Para cada $x \in \mathcal{N}$ y $k > 0$ tenemos que $q_k^T x > 0$ (por (1.16)), y

$$\begin{aligned} 0 \leq f(x)x \leq Ax \leq A_k x \quad &\Longrightarrow \quad f(x)q_k^T x \leq q_k^T A_k x = r_k q_k^T x \quad \Longrightarrow \quad f(x) \leq r_k \\ &\Longrightarrow \quad f(x) \leq r \quad (\text{porque } r_k \to r^\star = r). \end{aligned}$$

Como $f(z) = r$ y $z \in \mathcal{N}$, se sigue que $\max_{x\in\mathcal{N}} f(x) = r$. □

1.3.3. Frobenius

Esto es cuanto podemos generalizar el teorema de Perron para matrices no negativas sin añadir hipótesis adicionales. Por ejemplo, la matriz $A = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ muestra que las propiedades 1, 3, y 4 del teorema de Perron para matrices positivas no se verifican en general para matrices no negativas, y $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ muestra que tampoco la propiedad 6 se conserva. Lejos de aceptar que las principales cuestiones relativas a las propiedades espectrales de las matrices no negativas habían sido resueltas, F.G. Frobenius tuvo la perspicacia en 1912 de mirar bajo la superficie. Frobenius observa que el problema no

deriva simplemente de la existencia de entradas nulas sino también de la posición de estas. Por ejemplo, las propiedades 3 y 4 del teorema de Perron no se verifican para

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \text{ pero son válidas para } B = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

La genialidad de Frobenius fue ver que la diferencia entre A y B se puede expresar en términos de matrices reducibles (o irreducibles) y relacionar esas ideas con las propiedades espectrales de las matrices no negativas. La siguiente sección introduce esas ideas.

1.3.4. Grafos y matrices irreducibles

Definición 1.18. Un *grafo* es un conjunto de nodos $\{N_1, N_2, \ldots, N_n\}$ y un conjunto de aristas $\{E_1, E_2, \ldots, E_k\}$ entre los nodos. Un *grafo conexo* es un grafo que para cualquier par de nodos existe una secuencia de aristas que los une. Por ejemplo, el grafo mostrado en el lado izquierdo de la figura 1.2 es no dirigido y conexo.

Un *grafo dirigido* o *digrafo* es un grafo que contiene aristas dirigidas. Un grafo dirigido se dice que es *fuertemente conexo* si para cada par de nodos (N_i, N_k) hay una secuencia de aristas dirigidas que van desde N_i a N_k. El grafo en el lado derecho de la figura 1.2 es dirigido pero *no* fuertemente conexo (p.e. no se puede ir desde N_3 a N_1).

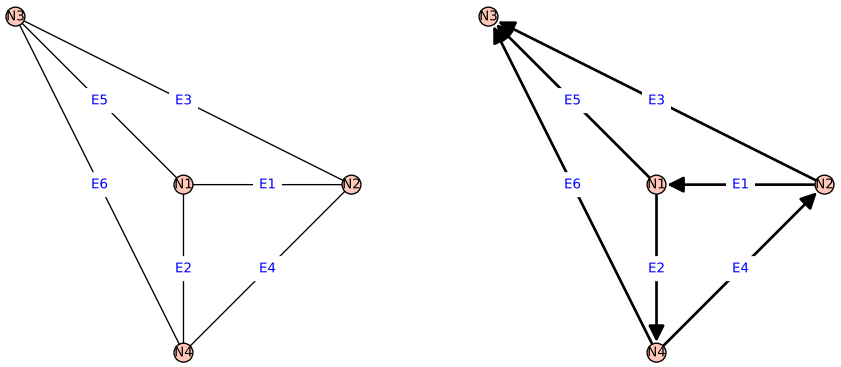


Figura 1.2: Dos ejemplos de grafos.

Cada grafo define dos útiles matrices, una *matriz de adyacencia* y una *matriz de incidencia*. Para un grafo $\mathcal{G}$ con nodos $\{N_1, N_2, \ldots, N_n\}$, la *matriz de adyacencia* $L_{n\times n}$ es la matriz con entradas

$$l_{ij} = \begin{cases} 1, & \text{si hay una arista de } N_i \text{ a } N_j, \\ 0, & \text{en caso contrario.} \end{cases}$$

Si $\mathcal{G}$ es no dirigido, entonces la matriz de adyacencia L es simétrica (i.e. $L = L^T$). Por ejemplo, las matrices de adyacencia para los grafos mostrados en la figura 1.2 son

$$L_1 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \qquad L_2 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Para un grafo no dirigido $\mathcal{G}$ con nodos $\{N_1, N_2, \ldots, N_n\}$ y aristas $\{E_1, E_2, \ldots, N_k\}$, *la matriz de incidencia* $C_{n\times k}$ tiene entradas

$$c_{ij} = \begin{cases} 1, & \text{si el nodo } N_i \text{ toca la arista } E_j, \\ 0, & \text{en caso contrario.} \end{cases}$$

Si $\mathcal{G}$ es un grafo dirigido, entonces su matriz de incidencia tiene entradas

$$c_{ij} = \begin{cases} 1, & \text{si la arista } E_j \text{ entra en el nodo } N_i, \\ -1, & \text{si la arista } E_j \text{ sale del nodo } N_i, \\ 0, & \text{si la arista } E_j \text{ ni empieza ni termina en el nodo } N_i. \end{cases}$$

Por ejemplo, las matrices de incidencia para los dos grafos mostrados en la figura 1.2 son

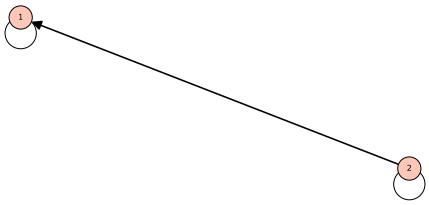
$$C_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \qquad C_2 = \begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & -1 \end{pmatrix}.$$

La siguiente proposición ([3, p. 203]) afirma que existe una relación directa entre la conexión de un grafo dirigido y el rango de su matriz de incidencia. Se entiende que un grafo dirigido es conexo cuando como grafo no dirigido es conexo.

Proposición 1.19. *Un grafo dirigido con n nodos y matriz de incidencia C es conexo si y sólo si*

$$\operatorname{rank}(C) = n - 1. \tag{1.22}$$

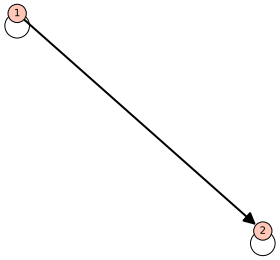
Hasta ahora hemos visto cómo construir una matriz a partir de un grafo, pero también podemos hacer el proceso a la inversa, i.e. construir un grafo a partir de una matriz. Dada una matriz $A_{n\times n}$, el grafo de A se define como el grafo *dirigido* $\mathcal{G}(A)$ con un conjunto de nodos $\{N_1, N_2, \ldots, N_n\}$ donde hay una arista dirigida de N_i *a* N_j si y sólo si $a_{ij} \neq 0$. Por ejemplo, si $A = \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix}$, entonces el grafo $\mathcal{G}(A)$ sería:



Cualquier producto de la forma P^TAP donde P es una matriz de permutación (una matriz obtenida de la matriz identidad I por permutación de sus filas o columnas) se llama *permutación simétrica* de A. Una matriz de permutación es ortogonal, i.e. $P^T = P^{-1}$. El efecto de una permutación simétrica en una matriz es el intercambio de filas en el mismo modo en que las columnas son intercambiadas. El efecto de una permutación simétrica en el grafo de una matriz es el reetiquetado de los nodos. En consecuencia, el grafo dirigido de una matriz es invariante a las permutaciones simétricas. Formalmente, $\mathcal{G}(P^TAP) = \mathcal{G}(A)$ siempre que P sea una matriz de permutación. Por ejemplo, si P es la matriz de permutación $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, y si usamos $A = \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix}$, entonces

$$P^TAP = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}, \tag{1.23}$$

y el gráfico $\mathcal{G}(P^TAP)$ sería:



Definición 1.20. Una matriz $A_{n\times n}$ se dice que es una *matriz reducible* cuando existe una matriz de permutación P tal que

$$P^TAP = \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}, \quad \text{donde } X \text{ y } Z \text{ son ambas matrices cuadradas.} \tag{1.24}$$

Por ejemplo, la matriz A en (1.23) es claramente reducible. La matriz $B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ también es reducible puesto que $P^TBP = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ para $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Naturalmente, una matriz *irreducible* es una matriz que no es reducible.

Veamos a continuación una útil caracterización para matrices irreducibles. La demostración se puede encontrar en [1, p. 402].

Teorema 1.21. *Sea $A_{n\times n} \geq 0$, entonces A es irreducible si y sólo si $(I + A)^{n-1} > 0$.*

Como muestra el siguiente teorema, los conceptos de matriz irreducible (o reducible) y conexión fuerte (o ausencia de ella) están íntimamente relacionados.

Teorema 1.22. *Sea A la matriz de adyacencia de un grafo dirigido. Entonces A es irreducible si y sólo si el grafo dirigido es fuertemente conexo. En otras palabras, A es irreducible si y sólo si para cada par de índices (i, j) hay una secuencia de entradas en A tal que $a_{ik_1}a_{k_1k_2}\cdots a_{k_tj} \neq 0$. Equivalentemente, A es irreducible si para cualquier matriz de permutación P,*

$$P^T AP \neq \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}, \quad \textit{donde } X \textit{ y } Z \textit{ son ambas matrices cuadradas.}$$

El precedente teorema nos proporciona un modo sencillo de determinar si una matriz es o no es reducible. Consideremos, por ejemplo, la matriz

$$A = \begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 9 & 2 & 0 & 4 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Sería un error tratar de usar la definición porque determinar si existe una matriz de permutación P tal que (1.24) se verifique probando con todas las matrices de permutación 5×5 es una tarea ardua. Sin embargo, examinando $\mathcal{G}(A)$ observamos en la figura 1.3 que es fuertemente conexo, así que A es irreducible.

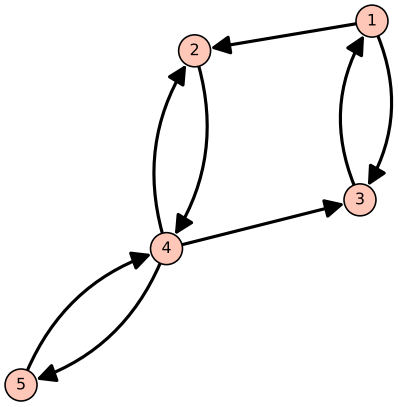


Figura 1.3: Grafo fuertemente conexo.

Para concluir este apartado veamos un ejemplo de matriz reducible. Usando la definición no es inmediato que la siguiente matriz sea reducible.

$$B = \begin{pmatrix} 0 & 0 & 0 & 2 & 0 & 0 \\ 8 & 0 & 0 & 1 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 7 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 \end{pmatrix}$$

Sin embargo, gracias al teorema 1.22 basta comprobar que $\mathcal{G}(B)$ no es un grafo fuertemente conexo, como se puede observar en la figura 1.4 (no podemos alcanzar el nodo 6 desde el nodo 4).

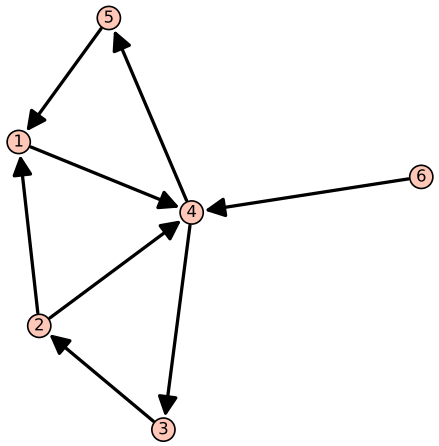


Figura 1.4: Grafo no fuertemente conexo.

1.3.5. El teorema de Perron-Frobenius

La contribución de Frobenius fue darse cuenta que el hecho de que las propiedades 1, 3, 4 y 6 del teorema de Perron para matrices positivas se perdiesen para matrices no negativas no derivaba sólo de la existencia de entradas nulas sino también de la localización de estas. En otras palabras, Frobenius entendió que las propiedades 1, 3, y 4 se mantienen si los ceros se encuentran en posiciones adecuadas, concretamente, las posiciones que aseguran que la matriz sea irreducible. Desafortunadamente, la irreducibilidad por sí sola no es suficiente para conservar la propiedad 6.

Enunciamos a continuación el teorema de Perron-Frobenius. Los detalles de la prueba pueden encontrarse en [3, p. 673].

Teorema 1.23. *Si $A_{n\times n} \geq 0$ es irreducible, entonces cada una de las siguientes afirmaciones es cierta.*

1. $r = \rho(A) > 0$.
2. $r \in \sigma(A)$ *(r es la raíz de Perron).*
3. $\text{alg mult}_A(r) = 1$ *(la raíz de Perron es simple).*
4. *Existe un autovector $x > 0$ tal que $Ax = rx$.*
5. *El vector de Perron es el único vector definido por*

$$Ap = rp, \qquad p > 0, \qquad \|p\|_1 = 1,$$

   y, a excepción de los múltiplos positivos de p, no hay otros autovectores no negativos para A, independientemente del autovalor.

6. *r no es necesariamente el único autovalor en la circunferencia espectral de A.*
7. $r = \text{máx}_{x\in\mathcal{N}} f(x)$, *(la fórmula de Collatz-Wielandt),*

$$\textit{donde } f(x) = \min_{\substack{1\leq i\leq n \\ x_i\neq 0}} \frac{(Ax)_i}{x_i} \quad y \quad \mathcal{N} = \{x \mid x \geq 0 \textit{ con } x \neq 0\}.$$

Ejemplo 1.24. Consideremos la matriz irreducible

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

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & -1 & -1 \\ -1 & \lambda & -1 \\ 0 & -1 & \lambda - 1 \end{vmatrix} = 0$$

$$\begin{aligned} \Longrightarrow \quad & \lambda^3 - 3\lambda^2 + 2 = 0 \\ \Longrightarrow \quad & (\lambda - 1)(\lambda^2 - 2\lambda - 2) = 0 \\ \Longrightarrow \quad & \lambda_1 = \sqrt{3} + 1, \quad \lambda_2 = 1 - \sqrt{3}, \quad \lambda_3 = 1 \end{aligned}$$

Vemos que $r = \lambda_1$ es el autovalor maximal de A. El autovector correspondiente al autovalor maximal se obtiene como solución del siguiente sistema lineal

$$\begin{pmatrix} \sqrt{3}-1 & -1 & -1 \\ -1 & \sqrt{3}+1 & -1 \\ 0 & -1 & \sqrt{3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Por tanto,

$$x = \begin{pmatrix} 2+\sqrt{3} \\ \sqrt{3} \\ 1 \end{pmatrix}$$

y se verifican las afirmaciones 1, 2, 3 y 4 del teorema 1.23.

1.3.6. Matrices primitivas

La única propiedad del teorema de Perron para matrices positivas que la irreducibilidad no conserva es la sexta, que afirma que existe un único autovalor en la circunferencia espectral. De hecho, $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ es no negativa e irreducible, pero los autovalores ± 1 están ambos en la circunferencia unitaria. La propiedad de tener (o no tener) un único autovalor en la circunferencia espectral divide el conjunto de matrices no negativas e irreducibles en dos importantes grupos.

Definición 1.25.

1. Una matriz A se dice que es una *matriz primitiva* cuando A es una matriz no negativa e irreducible que tiene un único autovalor, $r = \rho(A)$, en su circunferencia espectral.

2. Una matriz no negativa e irreducible con $h > 1$ autovalores en su circunferencia espectral se dice que no es *primitiva* (*imprimitiva* o *cíclica*), y h es llamado *índice de no primitividad* (*imprimitividad*).

3. Si A no es primitiva, entonces los h autovalores en su circunferencia espectral son

   $$\{r, r\omega, r\omega^2, \ldots, r\omega^{h-1}\}, \quad \text{donde} \quad \omega = e^{2\pi i/h} \text{ es una raíz } h\text{-primitiva de la unidad.}$$

   En otras palabras, estas son las raíces *h-ésimas* de $r^h = \rho(A)^h$, y están uniformemente espaciadas sobre la circunferencia. Además, cada autovalor $r\omega^k$ en la circunferencia espectral es simple.

Ejemplo 1.26. Consideremos la siguiente matriz

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda & -1 & 0 \\ 0 & \lambda & -1 \\ -1 & 0 & \lambda \end{vmatrix} = 0$$

$$\Longrightarrow \qquad \lambda^3 - 1 = 0$$

$$\Longrightarrow \qquad \lambda_1 = 1, \quad \lambda_2 = \omega, \quad \lambda_3 = \omega^2$$

donde $\omega = \frac{1}{2}(-1+i\sqrt{3})$. A tiene 3 autovalores de módulo $r = 1$ y por tanto es no primitiva. Su índice de no primitividad es 3.

Ejemplo 1.27. Consideremos otro ejemplo,

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda & -1 & -1 \\ -1 & \lambda & 0 \\ -1 & 0 & \lambda \end{vmatrix} = 0$$

$$\Longrightarrow \qquad \lambda^3 - 2\lambda = 0$$

$$\Longrightarrow \qquad \lambda_1 = \sqrt{2}, \quad \lambda_2 = -\sqrt{2}, \quad \lambda_3 = 0$$

A tiene 2 autovalores de módulo $r = \sqrt{2}$ y por tanto A es no primitiva. El índice de no primitividad de A es 2.

Veamos a continuación un teorema (véase [6, p. 54]) que interpreta en lenguaje de grafos dirigidos las matrices primitivas y no primitivas (cíclicas).

Teorema 1.28. *Sea $A_{n\times n} = (a_{ij}) \geq 0$ una matriz irreducible y denotemos por $\mathcal{G}(A)$ su grafo dirigido. Para cada nodo P_i de $\mathcal{G}(A)$, consideramos todos los caminos dirigidos y*

cerrados conectando P_i consigo mismo. Si S_i es el conjunto de todas las longitudes m_i de esos caminos dirigidos y cerrados, sea k_i el mayor común divisor de las longitudes, i.e.

$$k_i := \text{mcd}_{m_i \in S_i}\{m_i\}, \quad 1 \leq i \leq n.$$

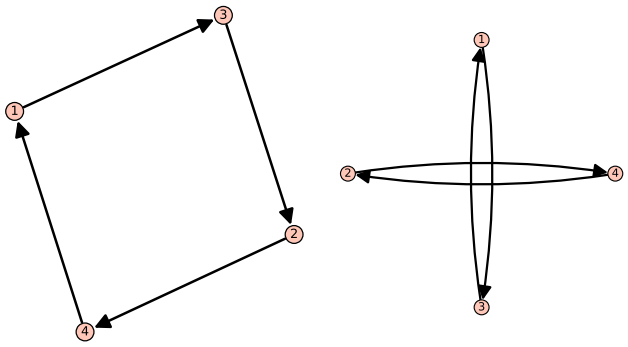
Entonces, $k_1 = k_2 = \cdots = k_n = k$, donde $k = 1$ cuando A es primitiva y $k > 1$ cuando A es imprimitiva (cíclica) de índice k.

Para ilustrar este resultado, veamos un ejemplo.

Ejemplo 1.29. Consideramos las siguientes matrices no negativas B_i definidas como

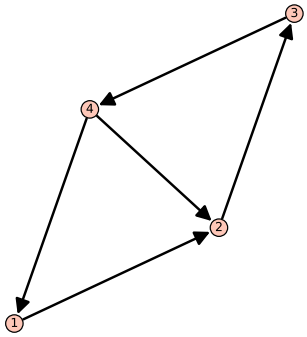
$$B_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, B_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, B_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}, B_4 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

cuyos grafos dirigidos son

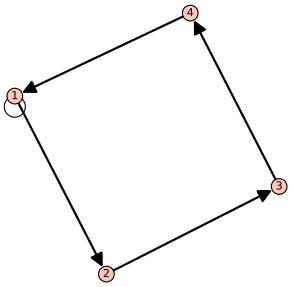


(a) $\mathcal{G}(B_1)$

(b) $\mathcal{G}(B_2)$



(c) $\mathcal{G}(B_3)$



(d) $\mathcal{G}(B_4)$

Figura 1.5: $\mathcal{G}(B_i)$

Observando los grafos dirigidos de las figuras 1.5a, 1.5b, 1.5c y 1.5d es fácil deducir que sus matrices son irreducibles, a excepción de la matriz B_2.

Ahora aplicamos el teorema 1.28 a las matrices B_1, B_3 y B_4. Para la matriz B_1, observando el grafo $\mathcal{G}(B_1)$ de la figura 1.5a se sigue que

$$k_1 = \text{mcd}\{4, 8, 12, \dots\} = 4,$$

por tanto B_1 es no primitiva (cíclica) de índice 4. Para la matriz B_3, tenemos caminos cerrados y dirigidos conectando P_4 consigo mismo de longitudes 3 y 4, cuyo máximo común divisor es la unidad. Por tanto, $k_4 = 1$ y concluimos que B_3 es primitiva. Para la matriz B_4, es obvio que hay un lazo, un camino cerrado y dirigido de longitud 1, conectando P_1 consigo mismo, así que $k_1 = 1$, y por tanto B_4 es también primitiva.

Las matrices primitivas tienen un sólo autovalor en la circunferencia espectral, veamos ahora cuál es su gran ventaja. Pues bien, la primitividad es lo que determina si las potencias de una matriz no negativa, irreducible y normalizada tienen un valor límite, y esto es una cuestión fundamental para la existencia del vector PageRank. La formulación precisa del teorema es la siguiente (véase [3, p. 674]).

Teorema 1.30. *Una matriz A no negativa e irreducible con $r = \rho(A)$ es primitiva si y sólo si* $\lim_{k\to\infty}(A/r)^k$ *existe, en cuyo caso*

$$\lim_{k\to\infty}\left(\frac{A}{r}\right)^k = \frac{pq^T}{q^Tp} > 0, \tag{1.25}$$

donde p y q^T son respectivamente los vectores de Perron por la derecha y por la izquierda de la matriz A.

Hemos visto que el método de la potencia es convergente para cualquier matriz diagonalizable, pero una matriz primitiva puede no ser diagonalizable. Gracias a este teorema obtenemos la convergencia del método de la potencia para cualquier matriz primitiva. Veamos un ejemplo realizado en SageMath de una matriz primitiva y no diagonalizable.

Ejemplo 1.31. Matriz primitiva y no diagonalizable.

```
A=matrix([[1/2,1/2,0,0],[0,0,1/2,1/2],[1/2,0,0,1/2],
[0,1/2,1/2,0]]);show(A)
```

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

```
I4 = identity_matrix(4)
```

```
show((I4+A)^3 )  # A es irreducible
```

$$\begin{pmatrix} \frac{7}{2} & \frac{5}{2} & 1 & 1 \\ 1 & 2 & \frac{5}{2} & \frac{5}{2} \\ \frac{5}{2} & \frac{13}{8} & 2 & \frac{15}{8} \\ 1 & \frac{15}{8} & \frac{5}{2} & \frac{21}{8} \end{pmatrix}$$

```
pc=A.charpoly();pc
```

```
x^4 - 1/2*x^3 - 1/2*x^2
```

```
solve(x^4 - 1/2*x^3 - 1/2*x^2==0, x,multiplicities=True)
# A es primitiva porque es irreducible y tiene un único \
   autovalor x=1 en su circunferencia espectral
```

```
([x == 1, x == (-1/2), x == 0], [1, 1, 2])
```

```
A.is_diagonalizable()
```

```
False
```

```
A.eigenspaces_right() # 0 tiene multiplicidad geométrica 1, \
   mientras que la algebraica es 2
```

```
[
(1, Vector space of degree 4 and dimension 1 over Rational Field
User basis matrix:
[1 1 1 1]),
(-1/2, Vector space of degree 4 and dimension 1 over Rational Field
User basis matrix:
[ 0  0  1 -1]),
(0, Vector space of degree 4 and dimension 1 over Rational Field
User basis matrix:
[ 1  -1 1 -1])
]
```

Si $A_{n\times n} \geq 0$ es irreducible pero no primitiva con $h > 1$ autovalores en la circunferencia espectral, entonces se puede demostrar [3] que cada uno de sus autovalores es simple y que están distribuidos uniformemente en la circunferencia espectral en el sentido que son las raíces *h-ésimas* de $r^h = \rho(A)^h$. Es decir, los autovalores en la circunferencia espectral están dados por

$$\{r, r\omega, r\omega^2, \ldots, r\omega^{h-1}\}, \quad \text{donde} \quad \omega = e^{2\pi i/h}.$$

Ahora bien, dada una matriz no negativa, ¿debemos calcular sus autovalores y contar cuántos caen en la circunferencia espectral para verificar la primitividad? La respuesta es negativa. Existen resultados que facilitan esta tarea.

Teorema 1.32. *Para una matriz A cuadrada y no negativa se verifican las siguientes afirmaciones.*

1. *A es primitiva si A es irreducible y tiene al menos un elemento diagonal positivo.*
2. *A es primitiva si y sólo si $A^m > 0$ para algún $m > 0$ (test de Frobenius para la primitividad [3, p. 678]).*

La primera afirmación proporciona una condición suficiente para la primitividad, mientras la segunda es una condición necesaria y suficiente. Por ejemplo, para determinar si la matriz $A = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix}$ es primitiva, el primer enunciado no es concluyente porque la diagonal de A está formada por ceros, así que nos vemos obligados a aplicar el segundo resultado. En vez de calcular las potencias de A, podemos simplificar el trabajo notando que si definimos la matriz booleana B como

$$b_{ij} = \begin{cases} 1, & \text{si } a_{ij} > 0, \\ 0, & \text{si } a_{ij} = 0, \end{cases}$$

entonces $[B^k]_{ij} > 0$ si y sólo si $[A^k]_{ij} > 0$ para cada $k > 0$. Por tanto, sólo necesitamos calcular las potencias de B (se puede demostrar que no es necesario el cálculo de más de $n^2 - 2n + 2$ potencias, gracias a un teorema de Wielandt que dice que A es primitiva si, y sólo si, $A^{(n-1)^2+1} > 0$). La matriz A del ejemplo es primitiva porque las potencias de B son

$$B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \; B^2 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \; B^3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \; B^4 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Aunque nosotros preferiríamos que nuestras matrices fuesen siempre primitivas, esto no es así en general. Por ello, resulta útil tener cierta compresión del índice de no primitividad. Las potencias de una matriz $A \geq 0$ irreducible nos dicen si A tiene más de un autovalor

en su circunferencia espectral, pero no ofrecen ninguna información sobre el número de autovalores. El siguiente teorema proporciona una opción teórica más simple que el cálculo real de todos los autovalores.

Teorema 1.33. *Si* $c(x) = x^n + c_{k_1}x^{n-k_1} + c_{k_2}x^{n-k_2} + \cdots + c_{k_s}x^{n-k_s}$ *es el polinomio característico de una matriz no primitiva* $A_{n\times n}$ *donde sólo aparecen los términos con coeficientes no nulos (i.e. cada* $c_{k_j} \neq 0$*, y* $n > (n-k_1) > \cdots > (n-k_s)$*), entonces el índice de no primitividad* h *es el máximo común divisor de* $\{k_1, k_2, \ldots, k_s\}$ *(véase [3, p. 679]).*

A veces es útil descomponer una matriz no primitiva, y la *forma de Frobenius* es la manera estándar para hacerlo.

Teorema 1.34. *Para cada matriz no primitiva* A *con índice de no primitividad* $h > 1$*, existe una matriz de permutación* P *tal que*

$$P^T AP = \begin{pmatrix} 0 & A_{12} & 0 & \cdots & 0 \\ 0 & 0 & A_{23} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & A_{h-1,h} \\ A_{h1} & 0 & \cdots & 0 & 0 \end{pmatrix}, \tag{1.26}$$

donde los bloques de ceros en la diagonal principal son cuadrados. Esta matriz es llamada la forma de Frobenius de una matriz no primitiva (véase [3, p. 680]).

Ejemplo 1.35. Consideremos la siguiente matriz,

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 0 & 3 \\ 0 & 4 & 0 \end{pmatrix}.$$

Los autovalores de A son

$$\lambda_1 = \sqrt{14}, \quad \lambda_2 = -\sqrt{14}, \quad \lambda_3 = 0.$$

Por tanto, el índice de no primitividad de A es 2. Consideremos la matriz de permutación

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

Con esta matriz de permutación, A es congruente con la matriz

$$B = PAP^T = \left(\begin{array}{c|cc} 0 & 2 & 3 \\ \hline 1 & 0 & 0 \\ 4 & 0 & 0 \end{array}\right).$$

Observamos que B está en forma de Frobenius con número de bloques igual al índice, i.e 2.

Para terminar esta sección incluimos una serie de ejemplos de matrices primitivas y no primitivas realizados con SageMath.

Ejemplo 1.36.

```
A=matrix([[0,1,0],[1,0,1],[0,1,0]])
```

```
show(A)
```

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

```
I3 = identity_matrix(3)
```

```
show(I3+A)
```

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

```
show((I3+A)^2) # A es irreducible
```

$$\begin{pmatrix} 2 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

```
A.eigenvectors_right()
```

```
[(0, [(1, 0, -1)], 1),
(-1.414213562373095?, [(1, -1.414213562373095?, 1)], 1),
(1.414213562373095?, [(1, 1.414213562373095?, 1)], 1)]
```

```
pc=A.charpoly();pc
```

```
x^3 - 2*x
```

```
solve(x^3-2*x==0,x, multiplicities=True)       # -1 es raíz \
   2-primitiva de la unidad
```

```
([x == -sqrt(2), x == sqrt(2), x == 0], [1, 1, 1])
```

```
A.is_primitive() #      su radio espectral es sqrt(2) y tiene \
   dos autovalores con módulo  sqrt(2)
```

```
False
```

```
P=matrix([[0,1,0],[1,0,0],[0,0,1]]);show(P)
```

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
F=P.transpose()*A*P;show(F) # forma normal de Frobenius
```

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

```
A.is_permutation_of(F, check=True)
```

```
(True, ((1,2), (1,2)))
```

Ejemplo 1.37.

```
B1=matrix([[0,1],[0,0]])
```

```
show(B1)
```

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

```
I2 = identity_matrix(2)
```

```
show(I2+B1) # B1 es reducible
```

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

```
pc1=B1.charpoly();pc1 # el radio espectral es r=0. No verifica\
    las propiedades 1, 3, 4 y 6 del Teorema de Perron
```

```
x^2
```

Ejemplo 1.38.

```
show(I2+I2) # I2 es reducible
```

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Ejemplo 1.39.

```
B2=matrix([[0,1],[1,0]]); show(B2)
```

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

```
show(B2+I2) # B2 es irreducible
```

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

```
B2.is_primitive()
```

```
False
```

```
pc2=B2.charpoly();pc2
```

```
x^2 - 1
```

```
solve(x^2-1==0,x, multiplicities=True)          # -1 es raíz \
   2-primitiva de la unidad. No verifica la propiedad 6 del \
   Teorema de Perron
```

```
([x == -1, x == 1], [1, 1])
```

Ejemplo 1.40.

```
M=matrix([[0,-1,0,-1],[6,0,3,0],[0,2,0,2],[6,0,3,0]])
```

```
M.eigenvectors_right()
```

```
[(0, [(1, 0, -2, 0),
(0, 1, 0, -1)], 4)]
```

```
P=matrix([[0,0,0,1],[0,1,0,0],[0,0,1,0],[1,0,0,0]])
```

```
show(P)
```

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
show(P.transpose()*M*P) # forma normal de Frobenius
```

$$\begin{pmatrix} 0 & 0 & 3 & 6 \\ 0 & 0 & 3 & 6 \\ 2 & 2 & 0 & 0 \\ -1 & -1 & 0 & 0 \end{pmatrix}$$

```
M.is_primitive()
```

```
False
```

```
pc=M.charpoly();pc
```

```
x^4
```

```
solve(x^4==0,x, multiplicities=True) # i es raíz 4-primitiva \
   de la unidad
```

```
([x == 0], [4])
```

Ejemplo 1.41.

```
A=matrix([[2,0,1],[0,0,1],[3,0,0]])
```

```
show(A)
```

$$\begin{pmatrix} 2 & 0 & 1 \\ 0 & 0 & 1 \\ 3 & 0 & 0 \end{pmatrix}$$

```
P=matrix([[0,1,0],[1,0,0],[0,0,1]]);show(P)
```

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
show(P.transpose()*A*P) # A es reducible
```

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 3 & 0 \end{pmatrix}$$

```
show((I3+A)^2 ) # A es reducible
```

$$\begin{pmatrix} 12 & 0 & 4 \\ 3 & 1 & 2 \\ 12 & 0 & 4 \end{pmatrix}$$

Ejemplo 1.42.

```
A=matrix([[2,1,1],[1,0,1],[0,1,1]])
```

```
show((I3+A)^2)  # A es irreducible
```

$$\begin{pmatrix} 10 & 5 & 6 \\ 4 & 3 & 4 \\ 1 & 3 & 5 \end{pmatrix}$$

```
A.is_primitive()
```

```
True
```

```
pc=A.charpoly();pc
x^3 - 3*x^2 + 2
```

```
solve(x^3 - 3*x^2 + 2==0,x, multiplicities=True)
([x == -sqrt(3) + 1, x == sqrt(3) + 1, x == 1], [1, 1, 1])
```

1.4. Teoría de cadenas de Markov

Veremos en el capítulo 3 que el vector PageRank de Google es el vector probabilidad estacionario de una cadena de Markov de tiempo discreto y estado finito. Para comprender y analizar esta afirmación, necesitamos familiarizarnos con los conceptos asociados a las cadenas de Markov. Empecemos con algunas definiciones.

Definición 1.43.

(a) Una *matriz estocástica* es una matriz no negativa $P_{n\times n}$ en que la suma de cada fila es igual a 1. Por ejemplo,

$$P = \begin{pmatrix} 1/2 & 1/4 & 0 & 1/4 \\ 1/4 & 1/2 & 1/8 & 1/8 \\ 0 & 0 & 1 & 0 \\ 3/8 & 1/8 & 0 & 1/2 \end{pmatrix}.$$

(b) Un *proceso estocástico* es un conjunto de variables aleatorias $\{X_t\}_{t=0}^{\infty}$ con un rango común $\{S_1, S_2, \ldots, S_n\}$, que es llamado el *espacio de estados* para el proceso. El parámetro t es pensado generalmente como el tiempo, y X_t representa el estado del proceso en el instante t. Por ejemplo, consideramos el proceso de navegar por la Web siguiendo enlaces sucesivamente para moverse de una página web a otra. El espacio de estados es el conjunto de todas las páginas web, y la variable aleatoria X_t es la página web siendo visitada en el tiempo t.

Para especificar que el tiempo es considerado en modo discreto y no continuo se usa la frase "proceso de *tiempo discreto*". Análogamente, se usa la frase "proceso de *estado finito*" cuando el espacio de estados es finito. Nosotros nos ceñiremos a procesos de tiempo discreto y estado finito.

(c) Una *cadena de Markov* es un proceso estocástico que satisface la *propiedad de Markov*

$$P(X_{t+1} = S_j \mid X_t = S_{i_t},\ X_{t-1} = S_{i_{t-1}}, \ldots, X_0 = S_{i_0}) = P(X_{t+1} = S_j \mid X_t = S_{i_t})$$

para cada $t = 0, 1, 2, \ldots$ La notación $P(E \mid F)$ denota la probabilidad condicional de que el suceso E ocurra sabiendo que ocurre el suceso F. Es decir, la propiedad de Markov asegura que el estado de la cadena al siguiente instante de tiempo depende sólo del estado actual y no de la historia pasada de la cadena. Por ejemplo, el proceso de navegar en la Web es una cadena de Markov siempre que la siguiente página que el surfista web visita no dependa de las páginas que él ha visitado en el pasado sino sólo de la página actual. En otros términos, si el surfista selecciona aleatoriamente un enlace en la página actual para cambiar a la siguiente, entonces el proceso es una cadena de Markov. Este tipo de cadena se denomina *paseo aleatorio* en la estructura de enlaces de la Web.

(d) La *probabilidad de transición* $p_{ij}(t) = P(X_t = S_j \mid X_{t-1} = S_i)$ es la probabilidad de estar en el estado S_j en el instante t sabiendo que la cadena está en el estado S_i en el instante $t-1$, o dicho en manera más sencilla, es la probabilidad de transición del estado S_i en el instante $t-1$ al estado S_j en el instante t.

(e) La *matriz de probabilidad de transición* $P_{n\times n}(t) = [p_{ij}(t)]$ es obviamente una matriz no negativa y la suma de cada fila es 1. En otras palabras, $P(t)$ es una matriz estocástica para cada t.

(f) Una *cadena de Markov estacionaria* es una cadena de Markov en que las probabilidades de transición no varían con el tiempo, i.e. $p_{ij}(t) = p_{ij}$ para todo t. Las cadenas estacionarias son conocidas también como *cadenas homogéneas*.

- En este caso la matriz de probabilidad de transición es una matriz estocástica constante $P = [p_{ij}]$. En lo sucesivo, asumiremos que nuestras matrices son estacionarias.
- De esta manera, cada cadena de Markov define una matriz estocástica, pero el contrario es también cierto. Cada matriz estocástica $P_{n\times n}$ define una cadena de Markov en n estados porque las entradas p_{ij} definen un conjunto de probabilidades de transición que pueden ser interpretadas como una cadena de Markov estacionaria en n estados.

(g) Una *cadena de Markov irreducible* es una cadena cuya matriz de probabilidad de transición P es una matriz irreducible. Una cadena se denomina *reducible* cuando P es una matriz reducible.

- Una *cadena de Markov periódica* es una cadena irreducible cuya matriz de probabilidad de transición P es una matriz no primitiva. Estas cadenas son llamadas periódicas porque cada estado puede ser ocupado sólo en momentos periódicos de tiempo, donde el período es el índice de no primitividad. Por ejemplo, consideramos una cadena cuyo índice de no primitividad sea $h = 3$. La forma de Frobenius (1.26) nos asegura que los estados pueden ser reetiquetados creando tres grupos de estados para los cuales la matriz de transición y sus potencias tienen la forma

$$P = \begin{pmatrix} 0 & \star & 0 \\ 0 & 0 & \star \\ \star & 0 & 0 \end{pmatrix}, P^2 = \begin{pmatrix} 0 & 0 & \star \\ \star & 0 & 0 \\ 0 & \star & 0 \end{pmatrix}, P^3 = \begin{pmatrix} \star & 0 & 0 \\ 0 & \star & 0 \\ 0 & 0 & \star \end{pmatrix}, P^4 = \begin{pmatrix} 0 & \star & 0 \\ 0 & 0 & \star \\ \star & 0 & 0 \end{pmatrix}, \ldots,$$

  donde este patrón continua indefinidamente. Si la cadena empieza en un estado del grupo i, sólo es posible volver a ocupar un estado del mismo grupo en un número de etapas múltiplo de h.

- Una *cadena de Markov aperiódica* (*no periódica*) es una cadena irreducible cuya matriz de probabilidad de transición P es una matriz primitiva.

(h) Un *vector de distribución de probabilidad* (o "vector probabilidad") se define como un vector fila no negativo $p^T = (p_1, p_2, \ldots, p_n)$ tal que $\sum_k p_k = 1$.

(i) Un *vector probabilidad estacionario* para una cadena de Markov cuya matriz de transición es P es un vector probabilidad π^T tal que $\pi^T P = \pi^T$.

(j) El *vector probabilidad en el k-ésimo paso* para una cadena de Markov en n estados se define como

$$p^T(k) = (p_1(k), p_2(k), \ldots, p_n(k)), \quad \text{donde} \quad p_j(k) = P(X_k = S_j).$$

En otras palabras, $p_j(k)$ es la probabilidad de estar en el *j-ésimo* estado después del *k-ésimo* paso, pero antes del paso *(k+1)-ésimo*.

(k) El *vector de distribución inicial* es

$$p^T(0) = (p_1(0), p_2(0), \ldots, p_n(0)), \quad \textit{donde} \quad p_j(0) = P(X_0 = S_j).$$

Es decir, $p_j(0)$ es la probabilidad de que la cadena empiece en S_j.

Para ilustrar estos conceptos, consideremos la web formada por tres páginas de la figura 1.6,

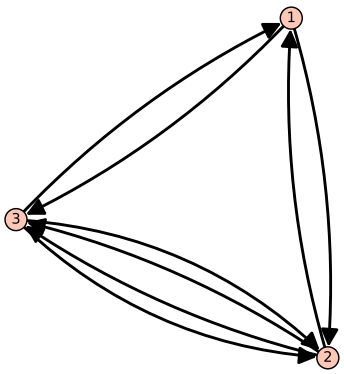


Figura 1.6: Web formada por 3 páginas

donde las aristas simbolizan los enlaces (p.e. la página 2 contiene dos enlaces a la página 3, y viceversa). La cadena de Markov definida por un paseo aleatorio en esta pequeña estructura de enlaces describe como un surfista web selecciona aleatoriamente un enlace en la página web en la que se encuentra. La matriz de transición para esta cadena es la matriz estocástica e irreducible

$$H = \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/3 & 0 & 2/3 \\ 1/3 & 2/3 & 0 \end{pmatrix}.$$

En este ejemplo, la matriz H es una matriz estocástica, pero si tuviésemos un nodo colgante (una nodo sin enlaces salientes), entonces H tendría una fila de ceros, en cuyo caso no sería estocástica y el proceso no sería una cadena de Markov. Como veremos más adelante, Google modifica la matriz de enlace para obtener siempre una matriz estocástica.

Si nuestro surfista web empieza en la página 2 en la figura 1.6, entonces el vector probabilidad inicial para la cadena es $p^T(0) = (0, 1, 0) = e_2^T$. Sin embargo, si el surfista selecciona aleatoriamente la página inicial, entonces $p^T(0) = (1/3, 1/3, 1/3) = e^T/3$ es el *vector de distribución uniforme*. Mediante el cálculo de autovalores habitual se obtiene que $\sigma(H) = \{1, -1/3, -2/3\}$ y por tanto H es una matriz no negativa con radio espectral $\rho(H) = 1$.

El hecho de que $\rho(H) = 1$ es una característica propia de las matrices estocásticas $P_{n\times n}$. Puesto que cada fila suma 1 entonces $\|P\|_\infty = 1$ o, equivalentemente, $Pe = e$, donde e es el vector columna formado sólo por unos. Como $(1, e)$ es un autopar para una matriz estocástica, y dado que se verifica $\rho(\star) \leq \|\star\|$ para cualquier norma matricial, se sigue que

$$1 \leq \rho(P) \leq \|P\|_\infty = 1 \quad \Longrightarrow \quad \rho(P) = 1. \tag{1.27}$$

Además, e es un autovector positivo asociado al autovalor $\rho(P) = 1$. No obstante, esto

no significa que e sea necesariamente el vector de Perron para P porque P puede no ser irreducible. Por ejemplo, la matriz $P = \begin{pmatrix} 1/2 & 1/2 \\ 0 & 1 \end{pmatrix}$ es estocástica y reducible. La necesidad de obtener la irreducibilidad es otra de las razones por las que Google modifica las filas de la matriz de enlace.

Capítulo 2

Introducción a los motores de búsqueda web

2.1. Breve historia de la búsqueda de información

La *búsqueda de información* es el proceso de buscar dentro de una colección de documentos una información deseada, llamada *consulta*. Desde el inicio de la humanidad existen colecciones de documentos, que van desde las pinturas realizadas en el interior de cuevas durante la Prehistoria hasta las colecciones de libros que conocemos en la actualidad. Diversos inventos a lo largo de la historia, tales como la invención de la imprenta en el siglo XVI o la difusión del papel a partir del 1800, provocaron que estas colecciones aumentasen enormemente. La búsqueda de información se fue haciendo cada vez más compleja y surgieron así rudimentarios sistemas de catalogación que organizaban los libros en función de su título o su autor usados principalmente en bibliotecas.

Estos inventos fueron progresando, pero la búsqueda no estaba todavía totalmente en manos del buscador de información. Fue necesaria la invención de la computadora digital entre el 1940 y el 1950 y las posteriores invenciones de los sistemas de búsqueda mediante ordenadores para acercarse al objetivo. El primer sistema de búsqueda por ordenador usaba una sintaxis especial para recuperar automáticamente información de libros y artículos relativa a la consulta del usuario. Desafortunadamente, la engorrosa sintaxis limitaba su dominio a bibliotecarios entrenados en su uso.

En 1989 el almacenaje, el acceso, y la búsqueda de información fueron revolucionados por un invento llamado World Wide Web, una colección de documentos repleta de enlaces. Sin embargo, los usuarios iniciales se encontraban con grandes dificultades cuando intentaban realizar una consulta. Los primeros motores de búsqueda web no hicieron mucho por

aliviar la frustración dado que devolvían miles de páginas web que el usuario debía abrir con el fin de determinar cuáles eran más relevantes para su consulta.

Todo esto cambió en 1998 cuando el *análisis de enlaces* entró en escena. Los motores de búsqueda más exitosos empezaron a usar el análisis de enlaces, una técnica que explota la información adicional inherente a la estructura vinculada de la Web para mejorar la calidad de los resultados. De esta forma, la búsqueda web mejoró drásticamente y los buscadores web promovieron sus motores favoritos como Google y AltaVista.

Hoy en día casi todos los principales motores de búsqueda combinan puntuaciones de análisis de enlaces junto con puntuaciones de búsqueda de información más tradicionales. En el siguiente capítulo estudiaremos el algoritmo PageRank, el sistema de análisis de enlaces empleado por Google.

2.2. La búsqueda de información en la Web

Con el nacimiento en 1989 de la World Wide Web, surge también una nueva rama dentro del mundo de la búsqueda de información que se especializa en las características particulares de esta colección de documentos.

La Web tiene una serie de propiedades que la diferencian de las colecciones tradicionales de documentos, por ejemplo, su enorme tamaño. Otra de sus particularidades es su doble dinamismo. Los libros tradicionales no cambian su contenido por sí mismos, sin embargo, las páginas web si lo hacen y con mucha frecuencia. Por otro lado, mientras que las colecciones tradicionales tienen un tamaño relativamente estático, miles de millones de páginas son añadidas a la Web cada año. Este dinamismo hace que sea difícil calcular las puntuaciones de relevancia para una consulta dada, puesto que la colección se encuentra en constante evolución.

En tercer lugar, la Web es auto-organizada. Mientras que las colecciones tradicionales de documentos están catalogadas por especialistas, cualquiera puede crear una página web y enlazar páginas a voluntad. Los datos son volátiles: hay actualizaciones, enlaces que se rompen y ficheros que desaparecen. Además, esta auto-organización abre la puerta a furtivos *spammers*, personas que usan técnicas de creación de páginas web engañosas para aumentar la puntuación de sus páginas en la clasificación de los buscadores web.

Otra característica particular de la Web es que está hipervinculada. Esta propiedad es clave en la eficiencia de los motores de búsqueda web, que explotan la información adicional disponible en la estructura de enlaces de la Web. Sin embargo, esta estructura hipervinculada tiene un efecto secundario negativo: los spammers diseñan estrategias de hipervínculo con el fin de incrementar el tráfico a sus páginas web.

Un último desafío de la búsqueda web concierne a la precisión. Como hemos comentado, la cantidad de información accesible no para de crecer, y sin embargo la habilidad del usuario para revisar los documentos devueltos no lo hace. Esto conlleva que la precisión de los motores de búsqueda debe incrementarse tan rápidamente como el número de documentos.

2.3. Los elementos del proceso de búsqueda web

En esta sección describiremos brevemente los elementos que participan en el sistema de búsqueda web con el fin de contextualizar el proceso de clasificación de las páginas web, que abordaremos en el siguiente capítulo. Véase [2] para profundizar en el proceso de búsqueda web.

- *Módulo de rastreo.* Como hemos visto en la sección anterior, la Web es auto-organizada y por tanto deben ser los motores de búsqueda los que realicen las tareas de recopilación y clasificación de datos. El módulo de rastreo se encarga de esta función utilizando unos robots virtuales llamados *arañas*, que recorren constantemente la Web reuniendo nuevas páginas web y almacenándolas en un repositorio central.

- *Página repositorio.* Almacena las páginas web completas.

- *Módulo de indexado.* Extrae la información vital de cada página web, creando una descripción comprimida de la página que es almacenada en varios índices.

- *Índices.* Los índices almacenan la información comprimida para cada página web. Destacan dos tipos de índices: el *índice de contenido*, que almacena el contenido (palabras clave, título ...), y el *índice de estructura*, que almacena información relativa a la estructura de hiperenlace de la Web.

Los cuatro módulos enumerados hasta ahora operan independientemente del usuario y de sus consultas, al contrario que los dos siguientes:

- *Módulo de consulta.* Este módulo traduce la consulta en lenguaje natural del usuario a un lenguaje que el sistema de búsqueda pueda entender y revisa los distintos índices para responder a la petición. Por ejemplo, consulta el índice de contenido para determinar que páginas usan los términos de la petición. Estas páginas son llamadas *páginas relevantes* y son enviadas al módulo de clasificación.

- *Módulo de clasificación.* Este módulo clasifica el conjunto de páginas relevantes basándose en ciertos criterios y devuelve una lista ordenada de páginas web donde las

páginas situadas en cabeza son aquellas que tienen mayor probabilidad de ser lo que el usuario desea. El módulo de clasificación cumple una función fundamental dado que el número de páginas relevantes que el usuario debe revisar es enorme. Con frecuencia, los filtros tradicionales de búsqueda de información no descartan suficientes páginas irrelevantes, por ello, la habitual *puntuación de contenido* se combina con una *puntuación de popularidad*. Se usan diversas reglas para dar a cada página una puntuación de contenido. Por ejemplo, muchos motores web dan una mayor puntuación a las páginas que contienen los términos de consulta en el título que a aquellas que los contienen en el cuerpo de la página. La puntuación de popularidad, que es el objetivo del siguiente capítulo, se determina a partir de un análisis de la estructura de hiperenlace de la Web. Ambas puntuaciones se combinan para determinar una valoración general para cada página relevante.

Capítulo 3

Algoritmo PageRank

3.1. Clasificación de páginas web según su popularidad

En 1998, la puntuación tradicional de contenido se tambaleaba debido al tamaño masivo de la Web y al acoso de los spammers. Afortunadamente, aparece la puntuación de popularidad para rescatarla. La puntuación de popularidad, también conocida como puntuación de importancia, aprovecha la información disponible en el inmenso grafo creado por la estructura de hiperenlace de la Web. Por ello, los modelos que explotan la estructura de hiperenlace de la Web son llamados *modelos de análisis de enlaces*. En este capítulo estudiaremos un clásico sistema de análisis de enlaces, el algoritmo PageRank.

El modelo PageRank surgió de la mente de dos doctorandos en informática de la Universidad de Stanford, Sergey Brin y Larry Page. Para comprender la idea intuitiva subyacente al algoritmo debemos entender un hiperenlace como una recomendación. Un hiperenlace de mi página web a la tuya es mi aprobación de tu página web. Por tanto, una página con más recomendaciones, materializadas a través de enlaces entrantes, debe ser más importante que una página con pocos enlaces entrantes. Sin embargo, igual que en una cita bibliográfica o en una carta de recomendación, el estatus de quien recomienda es también importante. Por otro lado, el peso de cada aprobación debe ser moderado por el número de recomendaciones hechas por quien recomienda.

En síntesis, la tesis del algoritmo PageRank es la siguiente: una página web es importante si es señalada por otras páginas importantes.

3.2. Fórmula original

En lo sucesivo consideraremos siempre autovectores por la izquierda.

Ha llegado el momento de traducir las ideas presentadas en la sección precedente en

ecuaciones matemáticas. Esta traducción revela que las puntuaciones de importancia PageRank son en realidad los valores estacionarios de una enorme cadena de Markov, y en consecuencia, la teoría de cadenas de Markov explica interesantes características del modelo PageRank usado por Google.

Para llevar a cabo nuestro objetivo necesitamos definir la Web como un grafo dirigido. Es decir, los nodos en el grafo representan las páginas web y las aristas dirigidas representan los hiperenlaces.

Brin y Page comenzaron el modelo con una simple ecuación de suma. El PageRank de la página P_i, denotado por $r(P_i)$, es la suma de los PageRanks de todas las páginas que enlazan con P_i.

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}, \tag{3.1}$$

donde B_{P_i} es el conjunto de páginas web que enlazan con P_i y $|P_j|$ es el *número de enlaces salientes de la página* P_j (*grado saliente de* P_j). Observamos que el PageRank de las páginas entrantes $r(P_j)$ en la ecuación (3.1) es moderado por el número de recomendaciones hechas por P_j, denotado por $|P_j|$. El problema de la ecuación (3.1) es que los valores $r(P_j)$, los PageRanks de las páginas que enlazan con la página P_i, son desconocidos. Para superar este problema, Brin y Page deciden recurrir a un proceso iterativo. Esto es, asumen que al inicio todas las páginas tienen igual PageRank ($1/n$, donde n es el número de páginas en el índice de Google de la Web). Ahora se utiliza la regla de la ecuación (3.1) para calcular $r(P_i)$ para cada página P_i en el índice. La ecuación (3.1) es aplicada sucesivamente, sustituyendo los valores de la iteración precedente en $r(P_j)$. Introducimos algunas notaciones para definir este *proceso iterativo*. Sea $r_{k+1}(P_i)$ el PageRank de la página P_i en la iteración $k+1$, entonces,

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}. \tag{3.2}$$

Este proceso es iniciado con $r_0(P_i) = 1/n$ para todas las páginas P_i y repetido con la esperanza de que las puntuaciones PageRank convergerán a unos valores finales estables. Aplicando la ecuación (3.2) al grafo de la figura 3.1 obtenemos los siguientes valores para los PageRanks después de unas pocas iteraciones.

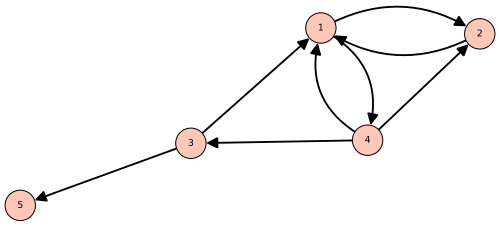


Figura 3.1: Grafo dirigido representando cinco páginas web.

Iteración 0	Iteración 1	Iteración 2	Rango en la Iter. 2
$r_0(P_1) = 1/5$	$r_1(P_1) = 11/30$	$r_2(P_1) = 7/30$	1
$r_0(P_2) = 1/5$	$r_1(P_2) = 1/6$	$r_2(P_2) = 13/60$	2
$r_0(P_3) = 1/5$	$r_1(P_3) = 1/15$	$r_2(P_3) = 1/30$	4
$r_0(P_4) = 1/5$	$r_1(P_4) = 1/10$	$r_2(P_4) = 11/60$	3
$r_0(P_5) = 1/5$	$r_1(P_5) = 1/10$	$r_2(P_5) = 1/30$	4

3.3. Representación matricial de la fórmula

Las ecuaciones (3.1) y (3.2) calculan el PageRank de las páginas una a una. Usando *matrices* podemos prescindir de los sumatorios, y a cada iteración, calcular un *vector* PageRank que contenga los valores de importancia para todas las páginas del índice. Con este fin introducimos una matriz H de dimensión $n \times n$ y un vector fila π^T de dimensión $1 \times n$. La matriz H es una matriz de *hiperenlace* normalizada por filas donde $H_{ij} = 1/|P_i|$ si hay un enlace desde el nodo i al nodo j, y 0, en caso contrario. Aunque H tiene el mismo número de elementos no nulos que la *matriz binaria de adyacencia* del grafo, sus elementos no nulos son probabilidades.

La matriz H para el grafo de la figura 3.1 es

$$H = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Los elementos no nulos de la fila i se corresponden con las páginas a las que enlaza la página i, mientras que los elementos no nulos de la columna i se corresponden con las páginas que llegan a la página i. Ahora introducimos un vector fila $\pi^{(k)T}$, que es el vector PageRank en la *k-ésima* iteración. Usando notación matricial, la ecuación (3.2) se puede

escribir en manera compacta como

$$\pi^{(k+1)T} = \pi^{(k)T} H. \tag{3.3}$$

Comentamos algunas observaciones acerca de la ecuación (3.3).

1. Cada iteración de la ecuación (3.3) involucra una multiplicación vector-matriz, que en general requiere $\mathcal{O}(n^2)$ *operaciones*, donde n es el tamaño de la matriz cuadrada H.

2. H es una matriz muy *dispersa* porque la mayoría de páginas web enlazan sólo a unas pocas páginas. Las matrices dispersas son bienvenidas por varias razones. La primera es que requieren un espacio de almacenamiento mínimo, ya que existen sistemas de almacenaje específicos para matrices dispersas que almacenan sólo los elementos no nulos y su posición [4]. En segundo lugar, una multiplicación vector-matriz implicando una matriz dispersa requiere un esfuerzo mucho menor que las *densas* $\mathcal{O}(n^2)$ operaciones. De hecho, requiere $\mathcal{O}(nnz(H))$ operaciones, donde $nnz(H)$ es el número de elementos no nulos en H. Las estimaciones muestran que de media una página web tiene alrededor de 10 enlaces salientes, lo que significa que H tiene aproximadamente $10n$ elementos no nulos, que contrasta con los n^2 elementos no nulos de una matriz completamente densa. Esto significa que la multiplicación vector-matriz de la ecuación (3.3) reduce su esfuerzo a $\mathcal{O}(n)$.

3. El proceso iterativo de la ecuación (3.3) es el clásico *método de la potencia* aplicado a la matriz H.

4. H se parece a una *matriz de probabilidad de transición estocástica* para una *cadena de Markov*. Los *nodos colgantes* de la red, aquellos nodos sin enlaces salientes, crean filas de ceros en la matriz. En cambio todas las otras filas, que se corresponden con nodos no colgantes, crean filas estocásticas. Por tanto, H es llamada *subestocástica*.

Estas cuatro observaciones son importantes para el desarrollo y la ejecución del modelo PageRank, e incidiremos sobre ellas durante el capítulo. Por ahora, nos centraremos en analizar la ecuación matricial iterativa (3.3).

3.4. Problemas con el proceso iterativo

Cuando observamos la ecuación (3.3) surgen en la mente de un matemático varias preguntas.

- ¿El proceso iterativo continua indefinidamente o es convergente?
- ¿Bajo qué circunstancias o propiedades de H tenemos garantizada la convergencia?
- ¿Convergerá a un vector que tenga sentido en el contexto del problema PageRank?
- ¿Convergerá a un único vector o a múltiples vectores?
- ¿La convergencia depende del vector inicial $\pi^{(0)T}$?
- Si converge, ¿cuántas iteraciones son necesarias para la convergencia?

Responderemos a todas estas preguntas durante las siguientes secciones analizando cómo Brin y Page decidieron resolver los problemas que surgían entorno a la ecuación (3.3).

Originariamente, Brin y Page iniciaron su proceso iterativo con $\pi^{(0)T} = 1/n\ e^T$, donde e^T *es el vector fila formado sólo por unos.* Aplicando la ecuación (3.3) con este vector inicial, tropezaron con varios problemas. Surge, por ejemplo, el problema de las páginas *sumideros de rango*: existen páginas que acumulan más y más PageRank en cada iteración, monopolizando las puntuaciones. En el ejemplo de la figura 3.2, el nodo colgante 3 es un sumidero de rango.

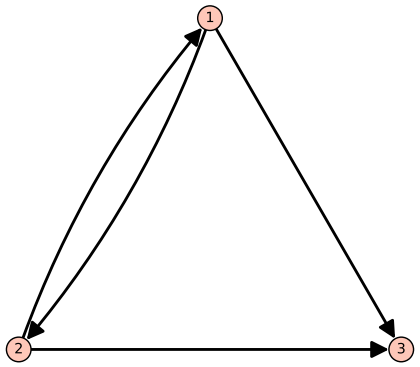


Figura 3.2: Grafo simple con sumidero de rango.

Debido a las páginas sumideros de rango puede ocurrir que unos nodos acumulen todo el PageRank, dejando a los restantes nodos con un PageRank igual a cero. En este caso, clasificar los nodos en función de sus valores PageRank resulta difícil pues la mayoría de los nodos están sujetos a un PageRank igual a cero. Por tanto, en una situación ideal el vector PageRank es positivo, i.e. contiene sólo valores positivos.

Los nodos colgantes pueden conducir también a un vector PageRank igual a cero. Consideremos el ejemplo de la figura 3.3.

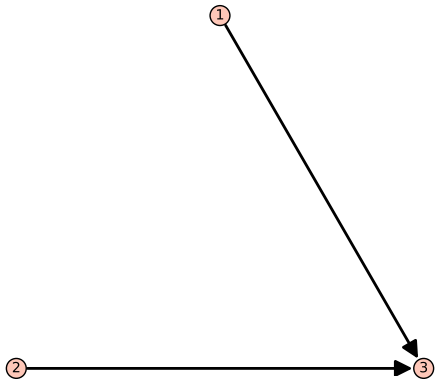


Figura 3.3: Grafo con nodo colgante.

Calculando el vector PageRank para este grafo obtenemos las siguientes iteraciones:

$$v_0^T = \begin{pmatrix} 1/3 & 1/3 & 1/3 \end{pmatrix}, \tag{3.4}$$

$$v_1^T = \begin{pmatrix} 1/3 & 1/3 & 1/3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 2/3 \end{pmatrix}, \tag{3.5}$$

$$v_2^T = \begin{pmatrix} 0 & 0 & 2/3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}. \tag{3.6}$$

Así que en este caso el rango de cada página es 0, pero esto es contraintuitivo pues la página 3 tiene dos enlaces entrantes por tanto debe tener alguna importancia.

Brin y Page se encontraron con una segunda dificultad, el problema de los *ciclos*. Consideremos el caso más simple de un ciclo representado en la figura 3.4.

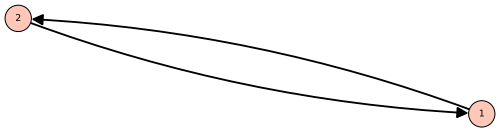


Figura 3.4: Grafo simple con ciclo.

La página 1 apunta a la página 2 y viceversa, creando un bucle o ciclo infinito. Supongamos que iniciamos el proceso iterativo de la ecuación (3.3) con $\pi^{(0)T} = (1 \quad 0)$. Entonces el algoritmo no convergerá nunca, los $\pi^{(k)T}$ oscilan indefinidamente entre $(1 \quad 0)$ cuando k es par y $(0 \quad 1)$ cuando k es impar.

3.5. Ajustes al modelo básico

En el punto 1.4 hemos estudiado los conceptos básicos relativos a las cadenas de Markov y ha llegado el momento de ponerlos en práctica. En las observaciones 3 y 4 (véase p. 48) comentamos que la ecuación (3.3) nos recuerda al método de la potencia aplicado a una cadena de Markov con *matriz de probabilidad de transición* H. Estas observaciones nos resultan muy útiles, pues la teoría de las cadenas de Markov está perfectamente desarrollada y es aplicable al problema PageRank. Gracias a la teoría de Markov podemos ajustar la ecuación (3.3) para conseguir resultados deseables. De acuerdo con lo visto en el capítulo preliminar, sabemos que para cualquier vector inicial, el método de la potencia aplicado a una matriz de Markov P converge a un único vector positivo llamado el *vector estacionario*, siempre que P sea *estocástica*, *irreducible* y *no periódica* (la no periodicidad sumada a la irreducibilidad implican la primitividad). Por tanto, los problemas de convergencia causados por páginas sumideros de rango y ciclos pueden ser resueltos si H es modificada de modo que verifique las propiedades antes mencionadas. En concreto, podemos afirmar que existe un único vector PageRank positivo si la matriz de Google es estocástica e irreducible. Si a esto le sumamos la no periodicidad, entonces el método de la potencia converge a este vector PageRank, independientemente del vector inicial usado en el proceso iterativo.

Por sorprendente que parezca, Brin y Page no usan las cadenas de Markov para explicar sus ajustes al modelo básico sino que se sirven de la noción de *surfista aleatorio*. Imaginemos un surfista web que rebota de una página a otra siguiendo los hipervínculos que conforman la estructura de la Web. Es decir, cuando él llega a un página con varios enlaces salientes, elige uno al azar y se traslada a la página siguiente, continuando este proceso aleatorio indefinidamente. A largo plazo, la proporción de tiempo que un surfista aleatorio pasa en una determinada página es una medida de la importancia relativa de esa página. Si él pasa una gran proporción de su tiempo en una página determinada, esto significa que siguiendo aleatoriamente hipervínculos se ha encontrado a sí mismo volviendo a esa página en repetidas ocasiones. Las páginas que el surfista visita con frecuencia deben ser importantes, porque son señaladas por otras páginas importantes.

Desafortunadamente, este surfista aleatorio se encuentra con algunos problemas. Por ejemplo, este queda atrapado cuando entra en un nodo colgante. Para solucionar esto, Brin y Page realizan un *ajuste de estocasticidad*: sustituyen las filas de ceros de H por $1/n\ e^T$, dando lugar a una matriz estocástica. Como resultado, después de entrar en un nodo colgante, el surfista aleatorio puede enlazar aleatoriamente a cualquier página web.

Para el ejemplo de la figura 3.1, la *matriz estocástica* que llamaremos S es

$$S = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix}.$$

Matemáticamente el ajuste de estocasticidad es $S = H + a(1/n\ e^T)$, donde $a_i = 1$ si la página i es un nodo colgante y 0 en caso contrario. El vector binario a es llamado *vector nodo colgante*. S es una combinación de las filas originales de la matriz de hiperenlace H y la matriz de rango uno $1/n\ ae^T$.

El ajuste realizado garantiza que S es estocástica, y por tanto, se trata de la matriz de probabilidad de transición para una cadena de Markov. Sin embargo, todavía no tenemos garantizados los resultados de convergencia deseados. Esto es, que exista un único π^T positivo y que la ecuación (3.3) converja a π^T rápidamente. Se necesita otro ajuste, un *ajuste de primitividad*, para conseguir así una matriz estocástica y *primitiva*. Una matriz primitiva es a la vez irreducible y aperiódica. Por tanto, el vector estacionario de la cadena (que es el vector PageRank en nuestro caso) existe, es único y el método de la potencia converge a él.

Para describir este ajuste de primitividad, Brin y Page recurren de nuevo a la noción de surfista aleatorio. Aunque es cierto que el surfista sigue la estructura de hiperenlace de la Web, es posible que se aburra y decida introducir una nueva URL en el navegador. Cuando esto ocurre, el surfista aleatorio se "teletransporta" a la nueva página, donde empieza a seguir hiperenlaces hasta que decide teletransportarse de nuevo. Para modelar matemáticamente este proceso, definimos una nueva matriz G como

$$G = \alpha S + (1 - \alpha)1/n\ ee^T,$$

donde α es un escalar entre 0 y 1. G es llamada la *matriz de Google*. En este modelo, α es un parámetro que controla la proporción de tiempo que el surfista aleatorio sigue los hiperenlaces en vez de teletransportarse. Supongamos que $\alpha = 0{,}4$. Entonces el 40 % del tiempo el surfista aleatorio sigue la estructura de hiperenlace de la Web y el otro 60 % del tiempo se teletransporta a una nueva página aleatoria. Podemos afirmar que el teletransporte es aleatorio porque la matriz de teletransporte $E = 1/n\ ee^T$ es uniforme, lo que significa que es igual de probable que salte a una página que a otra.

Existen varias consecuencias debidas al ajuste de primitividad.

- G es *estocástica*. Es la *combinación convexa* de dos matrices estocásticas S y $E = 1/n\ ee^T$.

- G es *irreducible*. Basta ver que el grafo de G es fuertemente conexo, pero esto es inmediato porque cada página está directamente conectada a cualquier otra página.

- G es *aperiódica*. Un estado es periódico si, partiendo de ese estado, sólo es posible volver a él en un número finito de etapas que sea múltiplo de un cierto número entero mayor que uno. En nuestra cadena de Markov uno puede volver a cada estado (esto es, a cada página) con probabilidad no nula al cabo de una etapa ($G_{ii} > 0$ para todo i) y por tanto los estados son aperiódicos.

- G es *primitiva* porque $G^k > 0$ para algún k (de hecho, esto es cierto para $k = 1$). Esta propiedad implica que existe un único vector positivo π^T, y el método de la potencia aplicado a G converge a este vector.

- G es completamente *densa*, lo cual es muy negativo en términos computacionales. Afortunadamente, G puede ser escrita como una *actualización de rango uno* de la matriz de hiperenlace H, que es muy dispersa. Como veremos en la siguiente sección, esto nos proporciona una gran ventaja.

$$\begin{aligned} G &= \alpha S + (1-\alpha)1/n\ ee^T \\ &= \alpha(H + 1/n\ ae^T) + (1-\alpha)1/n\ ee^T \\ &= \alpha H + (\alpha a + (1-\alpha)e)\ 1/n\ e^T. \end{aligned}$$

- G es artificial en el sentido de que las filas de la matriz de hiperenlace H han sido modificadas dos veces para obtener las propiedades de convergencia deseadas. Mientras que para la matriz H no tenemos garantizada la existencia del vector PageRank, para G existe un único vector PageRank, y resulta que este vector es muy bueno para puntuar las páginas web.

En resumen, el método PageRank ajustado es

$$\pi^{(k+1)T} = \pi^{(k)T} G, \tag{3.7}$$

que es sencillamente el *método de la potencia* aplicado a G.

Concluimos la sección con un ejemplo. Volviendo a la figura 3.1, para $\alpha = 0{,}85$, la matriz estocástica y primitiva G es

$$G = 0{,}85H + (0{,}85 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + 0{,}15 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}) \, 1/5 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 3/100 & 91/200 & 3/100 & 91/200 & 3/100 \\ 22/25 & 3/100 & 3/100 & 3/100 & 3/100 \\ 91/200 & 3/100 & 3/100 & 3/100 & 91/200 \\ 47/150 & 47/150 & 47/150 & 3/100 & 3/100 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix}.$$

El vector PageRank de Google es el vector estacionario de G y viene dado por

$$\pi^T = \begin{pmatrix} 0{,}3596132092 & 0{,}2538039380 & 0{,}1009683241 & 0{,}1977693023 & 0{,}0878452262 \end{pmatrix}.$$

La interpretación de $\pi_1 = 0{,}3596132092$ es que el $35{,}96132092\,\%$ del tiempo el surfista aleatorio visita la página 1. Por tanto, las páginas en este pequeño grafo web pueden ser ordenadas por sus importancia como $\begin{pmatrix} 1 & 2 & 4 & 3 & 5 \end{pmatrix}$, lo que significa que la página 1 es la más importante y la página 5 la menos importante.

3.6. Cálculo del vector PageRank

El problema PageRank puede ser interpretado en dos modos diversos:

1. Resolver el siguiente problema de *cálculo de autovectores* para π^T:

$$\pi^T = \pi^T G,$$
$$\pi^T e = 1.$$

2. Resolver el siguiente *sistema lineal homogéneo* para π^T:

$$\pi^T (I - G) = 0^T,$$
$$\pi^T e = 1.$$

En el primer sistema, el objetivo es encontrar el *autovector dominante normalizado por la izquierda* de G correspondiente al *autovalor dominante* $\lambda_1 = 1$ (G es una matriz

estocástica, por tanto $\lambda_1 = 1$). En el segundo sistema, el propósito es encontrar el vector normalizado perteneciente al espacio nulo por la izquierda de $I - G$. Ambos sistemas están sujetos a la ecuación de normalización $\pi^T e = 1$, que asegura que π^T es un vector de probabilidad. En el ejemplo de la sección 3.5, usamos SageMath para calcular el vector PageRank. Sin embargo, para una matriz del tamaño de la Web, como es el caso de la matriz de Google, este método es inviable. Como hemos visto, π^T es el vector estacionario de una cadena de Markov con matriz de transición G y existen numerosos métodos para calcular el vector estacionario para una cadena de Markov general [5]. Sin embargo, las características específicas de la matriz G hacen que el método de la potencia sea el más efectivo. En esta sección discutiremos los motivos que llevaron a Brin y Page a elegir el método de la potencia para calcular el vector PageRank.

El método de la potencia es uno de los métodos iterativos más antiguos y más simples para encontrar el *autovalor y el autovector dominante* de una matriz. El vector estacionario de una cadena de Markov es sencillamente el autovector dominante por la izquierda de la matriz de Markov, y por tanto, el método de la potencia puede ser utilizado para su cálculo. Sin embargo, si comparamos el método de la potencia con otros métodos iterativos disponibles, este es generalmente el más lento. Entonces, cabe preguntarse por qué Brin y Page lo eligieron. Existen varias razones que justifican su elección.

La primera es que el método de la potencia es simple. Su implementación y programación son elementales. Además, el método de la potencia aplicado a G (ecuación (3.7)) puede ser expresado en términos de la matriz dispersa H.

$$\begin{aligned}\pi^{(k+1)T} &= \pi^{(k)T} G \\ &= \alpha \pi^{(k)T} S + \frac{1-\alpha}{n} \pi^{(k)T}\, e\, e^T \\ &= \alpha \pi^{(k)T} H + (\alpha \pi^{(k)T} a + 1 - \alpha)\, e^T/n. \end{aligned} \tag{3.8}$$

Las multiplicaciones vector-matriz ($\pi^{(k)T} H$) son ejecutadas en la matriz extremadamente dispersa H, y las matrices S y G no están ni constituidas ni almacenadas, sólo sus componentes de rango uno, a y e, son necesarias. Recordemos que cada multiplicación vector-matriz requiere del orden de $\mathcal{O}(n)$ operaciones, dado que H tiene aproximadamente 10 elementos no nulos por fila. Esta es la razón principal que condujo a Brin y Page a usar el método de la potencia en 1998. Pero, ¿por qué el método de la potencia es todavía hoy el método predominante en los trabajos de investigación sobre el PageRank?, y ¿por qué la mayoría de las mejoras han sido modificaciones del método de la potencia en vez de experimentaciones con otros métodos? Las otras ventajas, que comentamos a continuación, responden a estas preguntas.

El método de la potencia, como muchos otros métodos iterativos, es *libre de matrices*. Esto significa que sólo se accede a los coeficientes matriciales a través de la rutina de multiplicación vector-matriz. No existe manipulación de la matriz, en contraste con los métodos directos, que manipulan los elementos de la matriz en cada iteración. En nuestro caso, almacenar los elementos de la matriz de Google no es factible. Incluso aunque H es una matriz muy dispersa, su enorme tamaño y su falta de estructura impiden el uso de métodos directos. En su lugar, se prefiere el uso de métodos libres de matrices.

El método de la potencia es también ventajoso con respeto al almacenamiento. Además de la matriz dispersa H y el vector nodo colgante a, sólo el iterando actual $\pi^{(k)T}$ debe ser almacenado. Este vector es completamente denso, lo que significa que n números reales deben ser almacenados. En el caso de Google $n = 8{,}1$ mil millones, lo que hace comprensible esta mentalidad de ahorro en el almacenamiento. Otros métodos iterativos, tales como GMRES, aunque son más rápidos, requieren el almacenamiento de varios vectores. Por ejemplo, GMRES requiere el almacenamiento de 10 vectores de longitud n en cada iteración, que es equivalente a la cantidad de almacenaje requerida para la matriz H.

La última razón que justifica el uso del método de la potencia para calcular el vector PageRank concierne al número de iteraciones requeridas. Brin y Page afirmaron en sus documentos de 1998, y otros lo han confirmado, que tan sólo son necesarias entre 50 y 100 iteraciones antes de obtener la convergencia. Recordemos que cada iteración del método de la potencia requiere un esfuerzo del orden de $\mathcal{O}(n)$, por tanto, es difícil encontrar un método que pueda competir con un esfuerzo de 50 $\mathcal{O}(n)$ iteraciones. Los algoritmos cuyo tiempo de ejecución y esfuerzo computacional son lineales para el tamaño del problema son poco comunes.

Pero, ¿por qué el método de la potencia aplicado a G requiere sólo 50 iteraciones para la convergencia? La respuesta viene dada por la teoría de las cadenas de Markov. En general, la *velocidad asintótica de convergencia* para el método de la potencia aplicado a una matriz depende de la relación entre los dos autovalores de mayor magnitud, denotados por λ_1 y λ_2. Precisamente, la velocidad asintótica de convergencia es la velocidad a la cual $|\lambda_2/\lambda_1|^k \to 0$. Para matrices estocásticas como G, $\lambda_1 = 1$, así que $|\lambda_2|$ gobierna la convergencia. Como G es además primitiva, $|\lambda_2| < 1$. En general, determinar λ_2 numéricamente requiere un esfuerzo computacional que no conviene asumir para obtener una aproximación de la velocidad asintótica de convergencia. Afortunadamente, para el problema PageRank, es fácil ver que si los respectivos espectros son $\sigma(S) = \{1, \mu_2, \ldots, \mu_n\}$ y $\sigma(G) = \{1, \lambda_2, \ldots, \lambda_n\}$, entonces

$$\lambda_k = \alpha\mu_k \quad \text{para} \quad k = 2, 3, \ldots, n.$$

La demostración de esta afirmación se proporciona al final del capítulo. Por otro lado, la estructura de enlace de la Web hace muy probable que $|\mu_2| = 1$ (o al menos $|\mu_2| \approx 1$),

lo que implica que $|\lambda_2(G)| = \alpha$ (o $|\lambda_2(G)| \approx \alpha$). Como resultado, el parámetro α explica la convergencia después de 50 iteraciones. En sus documentos, los fundadores de Google usaron $\alpha = 0{,}85$, y en el último informe, este es todavía el valor usado por Google. $\alpha^{50} = 0{,}85^{50} \approx 0{,}000296$, que implica que en la 50-ésima iteración uno puede esperar aproximadamente $2-3$ *cifras de precisión* en la aproximación del vector PageRank. Resulta que este grado de precisión es adecuado a las necesidades de clasificación de Google. Matemáticamente, podrían ser necesarias diez cifras de precisión para distinguir entre elementos del vector PageRank, pero cuando las puntuaciones PageRank se combinan con las puntuaciones de contenido no es necesaria una mayor precisión.

Teorema 3.1. *Para la matriz de Google* $G = \alpha S + (1-\alpha)\ 1/n\ ee^T$,

$$|\lambda_2(G)| \leq \alpha.$$

Para el caso $|\mu_2(S)| = 1$, $|\lambda_2(G)| = \alpha$. *Por tanto, la velocidad asintótica de convergencia del método de la potencia para el problema PageRank* (3.8) *es la velocidad a la que* $\alpha^k \to 0$.

Ahora podemos dar respuestas afirmativas a las cuestiones planteadas en la sección 3.4. Gracias a los ajustes de estocasticidad y primitividad, el método de la potencia aplicado a G converge a un único vector positivo llamado vector PageRank, con independencia del vector inicial. Además, para obtener puntuaciones PageRank con aproximadamente τ cifras de precisión, basta realizar $\tau/\log_{10}\alpha$ iteraciones.

3.7. Teorema sobre el espectro de la matriz de Google

En este capítulo, hemos definido la matriz de Google como $G = \alpha S + (1-\alpha)\ 1/n\ ee^T$. Sin embargo, podemos generalizar esta definición sustituyendo el factor $1/n\ ee^T$ por ev^T, donde $v^T > 0$ es un vector probabilidad. En esa sección presentamos el teorema para el segundo autovalor de esta matriz de Google más genérica.

Teorema 3.2. *Si el espectro de la matriz estocástica* S *es* $\{1, \lambda_2, \lambda_3, \ldots, \lambda_n\}$, *entonces el espectro de la matriz de Google* $G = \alpha S + (1-\alpha)\ ev^T$ *es* $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$, *donde* v^T *es un vector probabilidad.*

Demostración. Como S es estocástica, $(1,\ e)$ es un autopar de S. Sea $Q = (e \quad X)$ una matriz no singular que tiene el autovector e como primera columna. Sea $Q^{-1} = \begin{pmatrix} y^T \\ Y^T \end{pmatrix}$. Entonces $Q^{-1}Q = \begin{pmatrix} y^Te & y^TX \\ Y^Te & Y^TX \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & I \end{pmatrix}$, que proporciona dos útiles identidades,

$y^T e = 1$ e $Y^T e = 0$. Como resultado, la transformación semejante

$$Q^{-1}SQ = \begin{pmatrix} y^T e & y^T SX \\ Y^T e & Y^T SX \end{pmatrix} = \begin{pmatrix} 1 & y^T SX \\ 0 & Y^T SX \end{pmatrix}$$

revela que $Y^T SX$ contiene los restantes autovalores de S, $\lambda_2, \ldots \lambda_n$. Aplicando la transformación semejante a $G = \alpha S + (1-\alpha)ev^T$ resulta

$$\begin{aligned} Q^{-1}(\alpha S + (1-\alpha)ev^T)Q &= \alpha Q^{-1}SQ + (1-\alpha)Q^{-1}ev^T Q \\ &= \begin{pmatrix} \alpha & \alpha y^T SX \\ 0 & \alpha Y^T SX \end{pmatrix} + (1-\alpha)\begin{pmatrix} y^T e \\ Y^T e \end{pmatrix}(v^T e \quad v^T X) \\ &= \begin{pmatrix} \alpha & \alpha y^T SX \\ 0 & \alpha Y^T SX \end{pmatrix} + \begin{pmatrix} (1-\alpha) & (1-\alpha)v^T X \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & \alpha y^T SX + (1-\alpha)v^T X \\ 0 & \alpha Y^T SX \end{pmatrix}. \end{aligned}$$

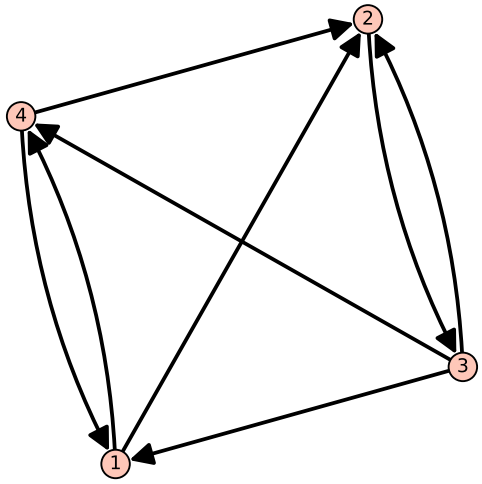
Por tanto, los autovalores de $G = \alpha S + (1-\alpha)ev^T$ son $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$. □

Capítulo 4

Ejemplos de cálculo del vector PageRank

En este capítulo incluimos algunos ejemplos de cálculo del vector PageRank realizados con SageMath. Recordemos que el vector PageRank es el único autovector probabilidad asociado al autovalor 1 de la matriz de Google. Pero antes de recurrir al modelo ajustado, veamos a qué conclusiones nos conduce un enfoque más básico.

Ejemplo 4.1. Consideremos una web que consista de cuatro páginas relacionadas entre sí de acuerdo con el siguiente esquema.



Este grafo fue representado usando el siguiente código en SageMath:

```
A = matrix([[0,1,0,1], [0,0,1,0],[1,1,0,1],[1,1,0,0]])
# donde A es la matriz de adyacencia del grafo.
```

```
show(A)
```

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

```
D=DiGraph(A)
```

```
D.relabel({0:'1' , 1:'2', 2:'3', 3:'4'})
```

```
D.plot()
```

Notamos que hemos reetiquetado los nombres de los vértices en el grafo de modo que la enumeración empiece en 1 y no en 0. Ahora que hemos creado la web, podemos fácilmente contar el número de enlaces entrantes en una página dada, basta sumar las entradas de la columna correspondiente:

```
n=A.ncols() # número de columnas de A
```

```
b=vector(0 for i in range(n))
```

```
for i in range(n):
        b[i]=sum(A[j][i] for j in range(n))
        # suma por columnas
```

```
print(b)
(2, 3, 1, 2)
```

El vector b contiene el número de enlaces entrantes en cada página web. Por tanto, la página web 2 parece ser la más importante porque tiene 3 enlaces entrantes, seguida por

las páginas 1 y 4, de igual importancia, y por último, la página 3.

No obstante, con este razonamiento no estamos teniendo en cuenta algunos importantes factores comentados en el capítulo anterior:

1. Un enlace de una página importante debería tener mayor valor que un enlace procedente de una página mediocre.
2. No todas las páginas web están conectadas entre sí, entonces, ¿cómo comparamos estas páginas?
3. El peso de cada recomendación (enlace) debería ser moderado por el número de recomendaciones hechas por la página que recomienda.
4. Algunas páginas podrían no tener ningún enlace saliente.

Ejemplo 4.2. Consideremos de nuevo el grafo del ejemplo 4.1. Sea x_i la puntuación de importancia de la página i, entonces las puntuaciones de importancia de las páginas web son las siguientes:

$$\begin{aligned} x_1 &= \frac{x_3}{3} + \frac{x_4}{2} \\ x_2 &= \frac{x_1}{2} + \frac{x_3}{3} + \frac{x_4}{2} \\ x_3 &= \frac{x_2}{1} \\ x_4 &= \frac{x_1}{2} + \frac{x_3}{3} \end{aligned}$$

Estas igualdades pueden ser escritas en forma matricial como $x^T = x^T H$, donde

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad \text{y} \quad H = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix}.$$

Por tanto, encontrar las puntuaciones de importancia de las páginas web es equivalente a encontrar el autovector por la izquierda de H asociado al autovalor 1. A continuación, realizamos estos cálculos con SageMath.

```
Ad = matrix([[0,1,0,1], [0,0,1,0],[1,1,0,1],[1,1,0,0]])
# matriz de adyacencia
```

```
D=DiGraph(A)
```

```
# Primero necesitamos normalizar la matriz de adyacencia Ad \
   para que la suma de las entradas de cada fila sea 1.
```

```
n=(Ad).ncols() # número de columnas de Ad
```

```
l=vector(0 for i in range(n))
```

```
for i in range(n):
        l[i]=sum([i,j] for j in range(n)) # contamos el número\
    de enlaces salientes de la página i
```

```
A0=matrix(n) # creamos una matriz nula de dimensión nxn
```

```
for i in range(n):
        A0[i,i]=l[i]
```

```
show(A0.inverse()*Ad)
```

$$\begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

```
H=(A0.inverse()*Ad)
```

```
show(H)
```

$$\begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

```
f=H.charpoly()
```

```
f.factor()
```

$(x-1)\cdot(x+\frac{1}{2})\cdot(x^2+\frac{1}{2}x+\frac{1}{6})$

```
H.eigenvalues()
```

$[1, -\frac{1}{2}, -0{,}25000000000000000? - 0{,}3227486121839514?\sqrt{-1}, -0{,}25000000000000000? + 0{,}3227486121839514?\sqrt{-1}]$

```
H.eigenvectors_left()
```

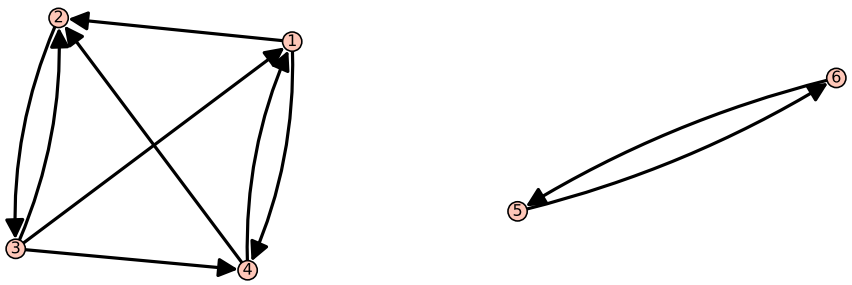
```
[(1, [(1, 3/2, 3/2, 1)], 1),
(-1/2, [(1, 0, 0, -1)], 1),
(-0.25000000000000000? - 0.3227486121839514?*sqrt(-1),
[(1, 0.25000000000000000? + 0.9682458365518542?*sqrt(-1), -2.2500000000000000? - 0.9682458365518542?*sqrt(-1), 1)], 1),
(-0.25000000000000000? + 0.3227486121839514?*sqrt(-1),

[(1, 0.25000000000000000? - 0.9682458365518542?*sqrt(-1), -2.2500000000000000? + 0.9682458365518542?*sqrt(-1), 1)], 1)]
```

Observamos que la función precedente nos proporciona primero el autovalor, luego su autovector asociado y por último su multiplicidad. Resulta que hemos obtenido un vector de importancias diferente:

- Enfoque básico: $(2, 3, 1, 2)$.
- Enfoque autovector: $(1, 3/2, 3/2, 1)$.

Sin embargo, si observamos de nuevo el digrafo que representa nuestra web, vemos que el nuevo resultado no es tan sorprendente: la página 2 recibe 3 votos y la página 3 sólo uno, pero este voto viene de la página 2 que es muy importante.

Ejemplo 4.3. Consideremos ahora una web compuesta de dos partes desconectadas, como es el caso del siguiente grafo:



La figura se ha realizado con el siguiente código:

```
D.add_path(['5','6'])
```

```
D.add_path(['6','5'])
```

```
D.plot()
```

Siguiendo el planteamiento realizado en el ejemplo 4.2, obtenemos exactamente las mismas relaciones para x_1, x_2, x_3, x_4, y además $x_5 = \frac{x_6}{1} = x_6$. Esto nos conduce a la ecuación matricial $x^T = x^T H$ con

$$H = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Como antes, calculamos el autovector por la izquierda de H asociado al autovalor 1 usando SageMath. Lo primero es calcular la matriz de adyacencia asociada al grafo.

```
Ad=D.adjacency_matrix()
```

```
show(Ad)
```

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

En segundo lugar calculamos la matriz de enlace.

```
n=Ad.ncols() # número de columnas de Ad
```

```
l=vector(0 for i in range(n))
```

```
for i in range(n):
        l[i]=sum(Ad[i,j] for j in range(n))
#contamos el número de enlaces salientes de la página i
```

```
D0=matrix(n) # matriz nula de dimensión nxn
```

```
for i in range(n):
        D0[i,i]=l[i]
```

```
H=(D0.inverse()*Ad)
```

```
show(H)
```

$$\begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

```
H.eigenvalues()
```

$[-\frac{1}{2}, -1, 1, 1, -0{,}25000000000000000? - 0{,}3227486121839514?\sqrt{-1}, -0{,}25000000000000000? + 0{,}3227486121839514?\sqrt{-1}]$

```
H.eigenvectors_left()
```

$[(-\frac{1}{2}, [(1, 0, 0, -1, 0, 0)], 1), (-1, [(0, 0, 0, 0, 1, -1)], 1),$
$(1, [(1, \frac{3}{2}, \frac{3}{2}, 1, 0, 0), (0, 0, 0, 0, 1, 1)], 2),$
$(-0{,}25000000000000000? - 0{,}3227486121839514?\sqrt{-1},$
$[(1, 0{,}25000000000000000? + 0{,}9682458365518542?\sqrt{-1}, -2{,}2500000000000000? - 0{,}9682458365518542?\sqrt{-1}, 1, 0, 0)], 1),$
$(-0{,}25000000000000000? + 0{,}3227486121839514?\sqrt{-1},$

$[(1, 0{,}25000000000000000? - 0{,}9682458365518542?\sqrt{-1}, -2{,}2500000000000000? + 0{,}9682458365518542?\sqrt{-1}, 1, 0, 0)], 1)]$

Vemos que el espacio asociado al autovalor 1 es de dimensión 2 y cualquier vector de la forma

$$\lambda \cdot (\frac{2}{10}, \frac{3}{10}, \frac{3}{10}, \frac{2}{10}, 0, 0)^T + (1-\lambda) \cdot (0, 0, 0, 0, \frac{1}{2}, \frac{1}{2})^T, \quad \lambda \in \mathbb{Q},$$

puede contener información sobre las puntuaciones de importancia de las páginas web.

Entonces, ¿cómo comparamos páginas web que no están conectadas mediante un camino de enlaces? Si existen páginas que no están conectadas mediante un camino de enlaces, esto significa que nuestro grafo es reducible. Por tanto debemos recurrir a lo aprendido en la sección 3.5 y realizar un ajuste de primitividad para obtener así una matriz G estocástica, irreducible y primitiva. Calculemos la matriz de Google G:

$$G = \alpha S + (1-\alpha)1/n\ ee^T.$$

De este modo, obtenemos una matriz G positiva y estocástica por filas, y por tanto, $\dim(\ker(G-I)) = 1$ (por ser estocástica $\rho(G) = 1$, y basta aplicar 1.12). Como en nuestro caso H es estocástica (no hay nodos colgantes), $S = H$.

Calculemos la matriz G con el siguiente código en SageMath:

```
H=matrix(QQ,[[0, 1/2,0, 1/2, 0, 0], [0, 0, 1, 0, 0, 0], [1/3, \
   1/3, 0, 1/3, 0, 0], [1/2, 1/2, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0]])
```

```
n=H.ncols()
```

```
alpha=0.85
```

```
E=ones_matrix(QQ,n)/n # matriz de unos de dimensión nxn \
   multiplicada por 1/n
```

```
G = alpha*H + (1-alpha)*E
```

```
show(G.n(digits=3)) # cada entrada tiene una precisión hasta \
   0.001 (tres cifras)
```

$$\begin{pmatrix} 0{,}0250 & 0{,}450 & 0{,}0250 & 0{,}450 & 0{,}0250 & 0{,}0250 \\ 0{,}0250 & 0{,}0250 & 0{,}875 & 0{,}0250 & 0{,}0250 & 0{,}0250 \\ 0{,}308 & 0{,}308 & 0{,}0250 & 0{,}308 & 0{,}0250 & 0{,}0250 \\ 0{,}450 & 0{,}450 & 0{,}0250 & 0{,}0250 & 0{,}0250 & 0{,}0250 \\ 0{,}0250 & 0{,}0250 & 0{,}0250 & 0{,}0250 & 0{,}0250 & 0{,}875 \\ 0{,}0250 & 0{,}0250 & 0{,}0250 & 0{,}0250 & 0{,}875 & 0{,}0250 \end{pmatrix}$$

Con el siguiente código podemos determinar en SageMath el autovector correspondiente al autovalor 1 para la matriz G.

```
H=matrix(QQ,[[0, 1/2,0, 1/2, 0, 0], [0, 0, 1, 0, 0, 0], [1/3, \
   1/3, 0, 1/3, 0, 0], [1/2, 1/2, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0]])
n=H.ncols()
alpha=0.85
E=ones_matrix(QQ,n)/n
G = alpha*H + (1-alpha)*E

for e,evs,k in G.change_ring(RDF).eigenvectors_left():
        if abs(e-1)<1e-6:
                if k==1:
                        ev = evs[0]
                        print(ev)
                else:
                        print("k not 1",k)
```

```
(0.33580009626152907, 0.47851513717267863, 0.4673944846341472,
0.33580009626152874, 0.4043774535824717, 0.4043774535824717)
```

Vemos que la página web más importante es la segunda y justo después la tercera, lo cual concuerda en cierta manera con el planteamiento del ejemplo 4.2. Gracias a SageMath no tuvimos que perder tiempo haciendo cálculos desagradables a mano. Sin embargo, cuanto más grande es la matriz, más tiempo tenemos que esperar la respuesta.

Ejemplo 4.4. Veamos ahora el modo más eficiente de calcular el vector PageRank cuando el tamaño de la matriz es muy grande: el método de la potencia. Recordemos el método:

$$\pi^{(k+1)T} = \alpha\pi^{(k)T}H + (\alpha\pi^{(k)T}a + 1 - \alpha)e^T/n.$$

Dado que nuestra matriz de enlace H es una matriz estocástica sin filas nulas, el vector nodo colgante a es un vector de ceros y la expresión anterior se simplifica:

$$\pi^{(k+1)T} = \alpha\pi^{(k)T}H + \frac{1-\alpha}{n}e^T.$$

El siguiente código implementa el método de la potencia en SageMath con vector inicial $x_0 = (1/n, \ldots, 1/n)$, donde n es el tamaño de la matriz H, y calcula 10 iteraciones.

```
H=matrix(QQ,[[0, 1/2,0, 1/2, 0, 0], [0, 0, 1, 0, 0, 0], [1/3, \
   1/3, 0, 1/3, 0, 0], [1/2, 1/2, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0]])
print('matriz original')
show(H)
H=matrix(RR,[[0, 1/2,0, 1/2, 0, 0], [0, 0, 1, 0, 0, 0], [1/3, \
   1/3, 0, 1/3, 0, 0], [1/2, 1/2, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0]])
n=H.ncols()
alpha=0.85
x0=vector(QQ,[1/n for i in range(n)])
N=10
E=ones_matrix(QQ,n)/n
G = alpha*H + (1-alpha)*E
print('matriz de Google')
show(G)
print('valor inicial', x0.n())
for k in range(N):
        x0=x0*G
        print( 'k = ', k, ':', x0)
```

matriz original

$$\begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

matriz de Google

$$\begin{pmatrix} 0{,}0250000000000000 & 0{,}450000000000000 & 0{,}0250000000000000 & 0{,}450000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 \\ 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}875000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 \\ 0{,}308333333333333 & 0{,}308333333333333 & 0{,}0250000000000000 & 0{,}308333333333333 & 0{,}0250000000000000 & 0{,}0250000000000000 \\ 0{,}450000000000000 & 0{,}450000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 \\ 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}875000000000000 \\ 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}0250000000000000 & 0{,}875000000000000 & 0{,}0250000000000000 \end{pmatrix}$$

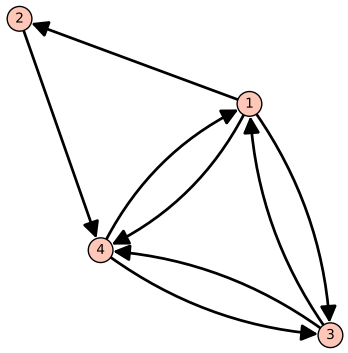
valor inicial (0.166666666666667, 0.166666666666667, 0.166666666666667, 0.166666666666667, 0.166666666666667, 0.166666666666667)

k = 0 : (0.143055555555556, 0.213888888888889, 0.166666666666667, 0.143055555555556, 0.166666666666667, 0.166666666666667)

```
k =  1 : (0.133020833333333, 0.193819444444445, 0.206805555555556,
0.133020833333333, 0.166666666666667, 0.166666666666667)
k =  2 : (0.140128761574074, 0.196662615740741, 0.189746527777778,
0.140128761574074, 0.166666666666667, 0.166666666666667)
k =  3 : (0.138316239872685, 0.197870963541667, 0.192163223379630,
0.138316239872685, 0.166666666666667, 0.166666666666667)
k =  4 : (0.138230648570120, 0.197015050516011, 0.193190319010417,
0.138230648570120, 0.166666666666667, 0.166666666666667)
k =  5 : (0.138485282695252, 0.197233308337553, 0.192462792938609,
0.138485282695252, 0.166666666666667, 0.166666666666667)
k =  6 : (0.138387369811422, 0.197243614956904, 0.192648312086920,
0.138387369811422, 0.166666666666667, 0.166666666666667)
k =  7 : (0.138398320594482, 0.197212952764336, 0.192657072713368,
0.138398320594482, 0.166666666666667, 0.166666666666667)
k =  8 : (0.138405456854776, 0.197224743107430, 0.192631009849686,
0.138405456854776, 0.166666666666667, 0.166666666666667)
k =  9 : (0.138401105287357, 0.197223424450637, 0.192641031641316,
0.138401105287357, 0.166666666666667, 0.166666666666667)
```

Observamos que en unas pocas iteraciones obtenemos el mismo orden de puntuaciones que en el ejemplo 4.3.

Ejemplo 4.5. Consideremos el siguiente grafo formado por cuatro páginas web:



En el siguiente código en SageMath vemos la evolución del vector PageRank cuando iniciamos el proceso iterativo con $x_0 = (1, 1, 1, 1)$.

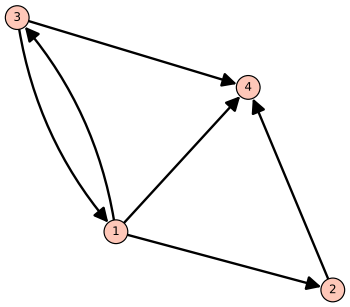
```
H=matrix(RR,[[0, 1/3,1/3, 1/3], [0, 0, 0, 1], [1/2, 0, 0, \
   1/2], [1/2, 0, 1/2, 0]])
n=H.ncols()
x0=vector(QQ,[1 for i in range(n)])
N=10
print('valor inicial', x0)
for k in range(N):
        x0=x0*H
        print('k = ', k, ':', x0)
x0=vector(QQ,[1 for i in range(n)])
```

```
valor inicial (1, 1, 1, 1)
k =  0 : (1.00000000000000, 0.333333333333333, 0.833333333333333, 1.83333333333333)
k =  1 : (1.33333333333333, 0.333333333333333, 1.25000000000000, 1.08333333333333)
k =  2 : (1.16666666666667, 0.444444444444444, 0.986111111111111, 1.40277777777778)
k =  3 : (1.19444444444444, 0.388888888888889, 1.09027777777778, 1.32638888888889)
k =  4 : (1.20833333333333, 0.398148148148148, 1.06134259259259, 1.33217592592593)
k =  5 : (1.19675925925926, 0.402777777777778, 1.06886574074074, 1.33159722222222)
k =  6 : (1.20023148148148, 0.398919753086420, 1.06471836419753, 1.33613040123457)
k =  7 : (1.20042438271605, 0.400077160493827, 1.06814236111111, 1.33135609567901)
k =  8 : (1.19974922839506, 0.400141460905350, 1.06581950874486, 1.33428980195473)
k =  9 : (1.20005465534979, 0.399916409465021, 1.06706131044239, 1.33296762474280)
```

Observamos que el algoritmo converge rápidamente en este ejemplo y resulta que la página 4 es la página con mayor puntuación. De hecho, la página 4 tiene 3 enlaces entrantes,

mientras que las otras páginas tienen sólo 1 o 2 enlaces entrantes. Esto es coherente con la lógica del algoritmo PageRank que nos dice que una página con mayor número de enlaces entrantes es más importante.

Ejemplo 4.6. En este ejemplo aplicaremos el algoritmo al siguiente grafo, cuyo nodo 4 es un nodo colgante.



De nuevo, iniciamos el proceso iterativo con $x_0 = (1, 1, 1, 1)$.

```
H=matrix(RR,[[0, 1/3,1/3, 1/3], [0, 0, 0, 1], [1/2, 0, 0, \
   1/2], [0, 0, 0, 0]])
n=H.ncols()
x0=vector(QQ,[1 for i in range(n)])
N=10
print('valor inicial', x0)
for k in range(N):
        x0=x0*H
        print('k = ', k, ':', x0)
x0=vector(QQ,[1 for i in range(n)])
valor inicial (1, 1, 1, 1)
k =  0 : (0.500000000000000, 0.333333333333333, 0.333333333333333,
1.83333333333333)
k =  1 : (0.166666666666667, 0.166666666666667, 0.166666666666667,
0.666666666666667)
k =  2 : (0.0833333333333333, 0.0555555555555556, 0.0555555555555556,
0.305555555555556)
k =  3 : (0.0277777777777778, 0.0277777777777778, 0.0277777777777778,
0.111111111111111)
k =  4 : (0.0138888888888889, 0.00925925925925926, 0.00925925925925926,
0.0509259259259259)
k =  5 : (0.00462962962962963, 0.00462962962962963, 0.00462962962962963,
```

```
0.0185185185185185)
k =  6 : (0.00231481481481481, 0.00154320987654321, 0.00154320987654321,
0.00848765432098765)
k =  7 : (0.000771604938271605, 0.000771604938271605, 0.000771604938271605,
0.00308641975308642)
k =  8 : (0.000385802469135802, 0.000257201646090535, 0.000257201646090535,
0.00141460905349794)
k =  9 : (0.000128600823045267, 0.000128600823045267, 0.000128600823045267,
0.000514403292181070)
```

Vemos que el vector PageRank convergerá finalmente al vector nulo. Para resolver este problema producido por la existencia de un nodo colgante necesitamos realizar un ajuste de estocasticidad (véase 3.5). Es decir, calculamos la matriz estocástica S definida como $S = H + a(1/n\ e^T)$, donde $a_i = 1$ si la página i es un nodo colgante y 0 en caso contrario. En nuestro caso,

$$S = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}.$$

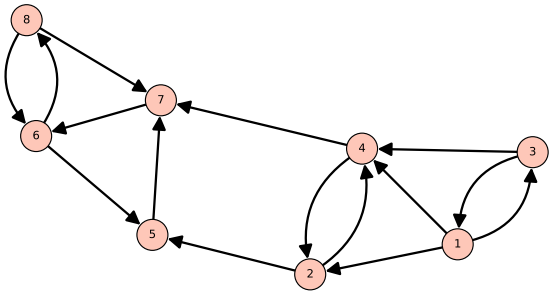
A continuación calculamos de nuevo la evolución del algoritmo PageRank aplicado a la matriz S.

```
S=matrix(RR,[[0, 1/3,1/3, 1/3], [0, 0, 0, 1], [1/2, 0, 0, \
   1/2], [1/4, 1/4, 1/4, 1/4]])
n=S.ncols()
x0=vector(QQ,[1 for i in range(n)])
N=10
print('valor inicial', x0)
for k in range(N):
        x0=x0*S
        print('k = ', k, ':', x0)
x0=vector(QQ,[1 for i in range(n)])
valor inicial (1, 1, 1, 1)
k =  0 : (0.750000000000000, 0.583333333333333, 0.583333333333333, 2.08333333333333)
k =  1 : (0.812500000000000, 0.770833333333333, 0.770833333333333, 1.64583333333333)
k =  2 : (0.796875000000000, 0.682291666666667, 0.682291666666667, 1.83854166666667)
k =  3 : (0.800781250000000, 0.725260416666667, 0.725260416666667, 1.74869791666667)
```

```
k =  4 : (0.799804687500000, 0.704101562500000, 0.704101562500000, 1.79199218750000)
k =  5 : (0.800048828125000, 0.714599609375000, 0.714599609375000, 1.77075195312500)
k =  6 : (0.799987792968750, 0.709370930989583, 0.709370930989583, 1.78127034505208)
k =  7 : (0.800003051757812, 0.711980183919271, 0.711980183919271, 1.77603658040365)
k =  8 : (0.799999237060547, 0.710676829020182, 0.710676829020182, 1.77864710489909)
k =  9 : (0.800000190734863, 0.711328188578288, 0.711328188578288, 1.77734343210856)
```

Observamos que gracias al ajuste de estocasticidad obtenemos un vector PageRank positivo, cuya página con mayor puntuación es la página 4.

Ejemplo 4.7. Consideremos el siguiente grafo formado por 8 páginas web:



En este grafo no tenemos nodos colgantes, sin embargo, un surfista web que llegue a las páginas 5, 6, 7 y 8 quedará atrapado en estas cuatro páginas. Si iniciamos el proceso iterativo con x_0 igual al vector fila de dimensión 1×8 con todas sus componentes iguales a $1/8$, entonces el algoritmo nos da las siguientes iteraciones.

```
H=matrix(RR,[[0, 1/3,1/3, 1/3, 0, 0, 0, 0], [0, 0, 0, 1/2, \
   1/2, 0, 0, 0], [1/2, 0, 0, 1/2, 0, 0, 0, 0], [0, 1/2, 0, 0,\
    0, 0, 1/2, 0], [0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 1/2,\
    0, 0, 1/2], [0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1/2,\
    1/2, 0]])
n=H.ncols()
x0=vector(QQ,[1/n for i in range(n)])
N=10
print('valor inicial', x0)
for k in range(N):
        x0=x0*H
        print('k = ', k, ':', x0)
x0=vector(QQ,[1 for i in range(n)])
```

```
valor inicial (1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8)

k =  0 : (0.0625000000000000, 0.104166666666667, 0.0416666666666667,
0.166666666666667, 0.125000000000000, 0.187500000000000, 0.250000000000000,
0.0625000000000000)
k =  1 : (0.0208333333333333, 0.104166666666667, 0.0208333333333333,
0.0937500000000000, 0.145833333333333, 0.281250000000000, 0.239583333333333,
0.0937500000000000)
k =  2 : (0.0104166666666667, 0.0538194444444444, 0.00694444444444444,
0.0694444444444444, 0.192708333333333, 0.286458333333333, 0.239583333333333,
0.140625000000000)
k =  3 : (0.00347222222222222, 0.0381944444444444, 0.00347222222222222,
0.0338541666666667, 0.170138888888889, 0.309895833333333, 0.297743055555556,
0.143229166666667)
k =  4 : (0.00173611111111111, 0.0180844907407407, 0.00115740740740741,
0.0219907407407407, 0.174045138888889, 0.369357638888889, 0.258680555555556,
0.154947916666667)
k =  5 : (0.000578703703703704, 0.0115740740740741, 0.000578703703703704,
0.0101996527777778, 0.193721064814815, 0.336154513888889, 0.262514467592593,
0.184678819444444)
k =  6 : (0.000289351851851852, 0.00529272762345679, 0.000192901234567901,
0.00626929012345679, 0.173864293981481, 0.354853877314815, 0.291160300925926,
0.168077256944444)
k =  7 : (0.000096450617283950 6, 0.00323109567901235, 0.000096450617283950 6,
0.00283926504629630, 0.180073302469136, 0.375198929398148, 0.261037567515432,
0.177426938657407)
k =  8 : (0.000048225308641975 3, 0.00145178272890946, 0.000032150205761316 9,
0.00169592335390947, 0.189215012538580, 0.349751036844136, 0.270206404320988,
0.187599464699074)
k =  9 : (0.000016075102880658 4, 0.000864036779835391, 0.000016075102880658 4,
0.000758041570216049, 0.175601409786523, 0.364006136670525, 0.283862706565072,
0.174875518422068)
```

Observamos que el rango de las páginas 1, 2, 3 y 4 converge a cero. Sin embargo, la página 4 tiene 3 enlaces entrantes y por tanto, debería tener una importancia no nula. Es decir, el algoritmo PageRank no funciona en este ejemplo y esto se debe a que este grafo no es irreducible. Podemos observar que no hay un camino uniendo el nodo 5 con el nodo 2, ni tampoco uniendo el nodo 7 con el nodo 4, por tanto, se trata de un grafo reducible.

Para calcular el vector PageRank para un grafo reducible debemos realizar un ajuste de primitividad (véase 3.5). Es decir, tenemos que recurrir a la matriz de Google definida por Page y Brin como: $G = \alpha S + (1-\alpha)1/n\ ee^T$. Como H es estocástica, $S = H$.

En el siguiente código calculamos la matriz G para $\alpha = 0,85$ y realizamos las 10 primeras iteraciones del método de la potencia aplicado a la matriz G.

```
H=matrix(QQ,[[0, 1/3,1/3, 1/3, 0, 0, 0, 0], [0, 0, 0, 1/2, \
   1/2, 0, 0, 0], [1/2, 0, 0, 1/2, 0, 0, 0, 0], [0, 1/2, 0, 0,\
    0, 0, 1/2, 0], [0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 1/2,\
    0, 0, 1/2], [0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1/2,\
    1/2, 0] ]);show(H)
```

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

```
H=matrix(RR,[[0, 1/3,1/3, 1/3, 0, 0, 0, 0], [0, 0, 0, 1/2, \
   1/2, 0, 0, 0], [1/2, 0, 0, 1/2, 0, 0, 0, 0], [0, 1/2, 0, 0,\
    0, 0, 1/2, 0], [0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 1/2,\
    0, 0, 1/2], [0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1/2,\
    1/2, 0]])
n=H.ncols()
alpha=0.85
x0=vector(QQ,[1/n for i in range(n)])
N=10
E=ones_matrix(QQ,n)/n
G = alpha*H + (1-alpha)*E
print('valor inicial', x0.n())
for k in range(N):
        x0=x0*G
        print('k = ', k, ':', x0)
```

```
valor inicial
(0.125000000000000, 0.125000000000000, 0.125000000000000, 0.125000000000000,
0.125000000000000, 0.125000000000000, 0.125000000000000, 0.125000000000000)
k =  0 :
(0.0718750000000000, 0.107291666666667, 0.0541666666666667,0.160416666666667,
 0.125000000000000, 0.178125000000000, 0.231250000000000, 0.0718750000000000)
k =  1 :
(0.0417708333333333, 0.107291666666667, 0.0391145833333333, 0.107734375000000,
0.140052083333333, 0.245859375000000, 0.223723958333333, 0.0944531250000000)
k =  2 :
(0.0353736979166667, 0.0763721788194445, 0.0305850694444444, 0.0928077256944445,
0.168839192708333, 0.249057942708333, 0.223723958333333, 0.123240234375000)
k =  3 :
(0.0317486545138889, 0.0682158311631945, 0.0287725477430556, 0.0742293782552084,
0.157057801649306, 0.261292464192708, 0.254083696831597, 0.124599625651042)
k =  4 :
(0.0309783327907986, 0.0592929378707321, 0.0277454521122685, 0.0689655131474248,
0.158791025526259, 0.287675983208550, 0.236751458062066, 0.129799297281901)
k =  5 :
(0.0305418171477141, 0.0568375373783818, 0.0275271942907263, 0.0645185100335015,
0.166211791458695, 0.275153440697564, 0.238197416129783, 0.141012292863634)
k =  6 :
(0.0304490575735587, 0.0548238816227571, 0.0274035148585190, 0.0632585258178899,
0.159846165682277, 0.281148028177360, 0.247380613971173, 0.135690212296465)
k =  7 :
(0.0303964938148706, 0.0542621064517782, 0.0273772329791750, 0.0623238764837173,
0.161538061665050, 0.286691862101495, 0.239172454528536, 0.138237911975378)
k =  8 :
(0.0303853240161494, 0.0538499874197932, 0.0273623399142133, 0.0620590591723684,
0.163655436635141, 0.280797698938791, 0.241296112510408, 0.140594041393135)
k =  9 :
(0.0303789944635407, 0.0537342752861656, 0.0273591751379090, 0.0618744142548618,
0.160975266702398, 0.283604163225929, 0.243984688880209, 0.138089022048986)
```

Vemos que ahora los rangos de las páginas 1, 2, 3 y 4 no tienden a cero.

Bibliografía

[1] R. A. Horn, C. R. Johnson, Matrix analysis, Cambridge University Press, 2012.

[2] A. N. Langville, C. D. Meyer, Google's PageRank and Beyond, Princeton University Press, 2006.

[3] C. D. Meyer, Matrix analysis and applied linear algebra, vol. 71, Siam, 2000.

[4] Y. Saad, Iterative methods for sparse linear systems, vol. 82, Siam, 2003.

[5] W. J. Stewart, Introduction to the numerical solution of Markov chains, Princeton University Press, 1994.

[6] R. S. Varga, Iterative analysis, Springer, 1962.