

Laboratorio de Métodos Numéricos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico Número 2

Ohhh solo tiran π -edras...

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gonzalo.ciruelos@gmail.com
Costa, Manuel José Joaquín	035/14	manuc94@hotmail.com
Gatti, Mathias Nicolás	477/14	mathigatti@gmail.com

asdf

key1 key2 key3 key4

Índice

1. Introducción teórica	3
2. Desarrollo	4
2.1. Convenciones	4
2.2. Métodos numéricos usados	4
2.2.1. Método de la potencia	4
2.2.2. PageRank	4
2.2.3. Método GeM	5
2.2.4. Otros métodos	6
2.3. Estructuración del código	6
2.3.1. Matriz	6
2.3.2. MatrizDep	7
2.3.3. Problema	7
2.4. Experimentación	8
3. Resultados y discusión	9
3.1. PageRank y páginas web	9
3.2. PageRank y ligas deportivas	10
4. Conclusiones	11
5. Apéndices	12
5.1. Propositiones	12
5.1.1. Proposición 1	12

1. Introducción teórica

El objetivo del presente informe es resolver un problema práctico mediante el modelado matemático del mismo. Este problema consiste en modelar páginas web y ligas deportivas con cadenas de Markov, con el objetivo de obtener rankings para ellas.

Para esto, dada una cadena de Markov, construida de una cierta manera que será formulada más adelante, se va a considerar el modelo de navegante aleatorio. En este modelo, se comienza en un nodo cualquiera del diagrama y se va navegando a través de los links. Además podemos extrapolar esta idea, originalmente diseñada para páginas web, y utilizarla en ligas deportivas.

Entonces, nuestro objetivo de alguna manera es rankear mejor aquellos nodos del diagrama de transición en los que el navegante aleatorio se encuentra más tiempo. Para ello, nuestro objetivo será encontrar un *estado estacionario*, dado que este representará cual es la probabilidad de que el navegante se encuentre en cada nodo, si lo dejaramos recorriendo el diagrama infinito tiempo.

Volviendo a las cadenas de Markov, si la matriz de transición de la cadena es P (o sea, P_{ij} es la probabilidad de pasar del estado i al j), estamos buscando algun x tal que

$$x^t P = x^t$$

O equivalentemente,

$$P^t x = x$$

Como P es una matriz de transición, sus filas son vectores de probabilidad, por lo que $0 \leq P_{ij} \leq 1$ y las filas suman 1. En consecuencia, como se prueba en [5.1.1](#), el autovalor de mayor módulo es 1.

2. Desarrollo

2.1. Convenciones

2.2. Métodos numéricos usados

Como dijimos en la introducción, nuestro objetivo será, dada una matriz de transición P , encontrarle un autovector de autovalor asociado igual a 1 a su transpuesta. (Usamos la transpuesta por comodidad notacional).

$$P^t x = x$$

2.2.1. Método de la potencia

El método de la potencia, dada una matriz A , produce un autovalor λ y un autovector asociado a λ , v no nulo. El método es iterativo, y se puede encontrar una mejor explicación sobre él en [DB74, Cap. 5.8.1].

El método consiste en tomar un $x^{(0)}$ inicial, y luego construir una sucesión $\{x^{(k)}\}$ de la siguiente manera:

$$x^{(i)} = \frac{Ax^{(i-1)}}{\|Ax^{(i-1)}\|}$$

Y entonces, bajo ciertas condiciones, si se toma k lo suficientemente grande, $x^{(k)} \rightarrow \bar{x}$, tal que $A\bar{x} = \lambda\bar{x}$, λ el autovalor de mayor módulo. Por ello establecemos como criterio de parada que la diferencia entre el vector generado en una iteración y su anterior sea lo suficientemente chica.

Como probamos en 5.1.1, el autovalor de máximo módulo en este caso es 1, pero puede pasar también que $\lambda = -1$ también sea un autovalor, pero comenzando con $x^{(0)} = (\frac{1}{n}, \dots, \frac{1}{n})$ inicial, nos aseguramos de que las entradas sean siempre positivas, consiguiendo así un autovector asociado a autovalor $\lambda = 1$.

2.2.2. PageRank

PageRank más que un método de cómputo es un método de modelado. Dado un grafo cuyos nodos representan páginas webs y sus aristas representan links entre las páginas web, nos permitirá modelar un navegante aleatorio utilizando una cadena de Markov. Los detalles de la construcción de la cadena y la matriz asociada pueden encontrarse en [BP98].

Proveeremos una breve explicación de como se arma la matriz de transición utilizando un vector fila de la matriz P . P_i es la i -ésima fila de la matriz, y su entrada j -ésima nos dice la probabilidad que habrá de ir de la página web i a la j . A priori una buena aproximación sería

$$P_{ij} = \begin{cases} \frac{1}{n_i} & \text{si hay un link de } i \text{ a } j \\ 0 & \text{si no} \end{cases}$$

Donde n_i es la cantidad de links salientes de la página i . El primer problema, obvio, es que en general, esta matriz no es de transición, porque si una página web no tiene links salientes, la matriz va a tener toda una fila de ceros. Por eso, en este caso, se agrega una fila que vale toda $(\frac{1}{n}, \dots, \frac{1}{n})$.

Luego, se introduce el concepto de teletransportación. La idea es que, con una cierta probabilidad $1 - c$, el navegante aleatorio puede saltar a cualquier página de toda la red sin importar en cual esté actualmente. Todo esto, nuevamente, está correctamente explicado en [BP98] y [Kam+03].

Luego, puede utilizarse el algoritmo descripto anteriormente para encontrar un auto-vector de autovalor asociado igual a 1.

En este trabajo en particular, utilizaremos una versión mejorada del método de la potencia, adaptada para este problema en particular, propuesta por [Kam+03]. Este consiste en separar el único paso del método de la potencia en 3 pasos separados, de tal manera de acelerar el cómputo, aprovechándonos de que la matriz de transición (sin agregarle el factor de teletransportación) es esparsa.

De esta manera, se pueden obtener importantes ganancias en lo que respecta a la performance.

2.2.3. Método GeM

El método GeM, propuesto en [GMA08], tiene como objetivo adaptar el algoritmo de PageRank para ligas deportivas. La idea es simple, al igual que en algoritmo original de PageRank, la idea es armar una cadena de Markov y modelar un navegante aleatorio.

En este modelo, se representa una temporada (o una fecha, o un periodo de tiempo cualquiera) como un grafo dirigido y pesado, al igual que en el modelo de PageRank. Sin embargo, en este caso, los pesos de la primera matriz no valen 0 o 1, si no que toman el valor del valor absoluto de los puntajes de cada partido.

De esta manera, si el equipo i perdió contra el equipo j por p puntos, en la primera matriz A , valdrá que $A_{ij} = p$.

Luego, al igual que en PageRank, las filas de esta matriz que valgan 0 (eso significa que el equipo está invicto hasta el momento) serán completadas y además se agregará el factor de teletransportación, haciendo que todas las entradas de la matriz P sean distintas de 0.

Al igual que antes, nuestro objetivo es encontrar un autovector de autovalor 1 para P^t , y para ello utilizaremos el método de la potencia común y corriente.

Es importante notar que este método es, a diferencia de el anterior, un método que nos indica *cómo* modelar, mientras que el método propuesto en [Kam+03] lo que hace es tomar un modelo conocido e intentar mejorar su velocidad de cómputo.

2.2.4. Otros métodos

Además de los métodos mencionados anteriormente, para cada problema (ranqueo de páginas web y de ligas deportivas), utilizaremos un método alternativo.

En el caso de las páginas webs será el conocido como IN-DEGREE, que rankea mejor a aquellas páginas que son mas linkeadas y peor a las que son menos linkeadas. En la parte de experimentación se comparará estos métodos con la requerida profundidad.

En el caso de los rankings de ligas deportivas, utilizaremos el método standard de ranqueo de cada deporte. En el caso del fútbol, este consiste en caso de empate un punto a cada equipo, y en otro caso darle 3 puntos al equipo ganador y 0 al perdedor.

2.3. Estructuración del código

Para el modelado del problema diseñamos tres módulos: Matriz, MatrizDep y Problema.

2.3.1. Matriz

El módulo Matriz es el que se usará para representar a las matrices de conectividad de redes de páginas web.

Representación interna Como la matriz de conectividad es en general esparsa (dado que cada página web se conecta en proporción con muy pocas páginas), es conveniente utilizar una representación que aproveche esto. Para eso, vamos a usar una representación conocida como Compressed Row Storage (CRS). Para más información sobre este formato puede consultarse [Bar+].

Elegimos este formato porque será especialmente cómodo a la hora de hacer el producto iterativo del método de la potencia, dado que si queremos hacer $P^t x$, es conveniente poder acceder a P^t por filas fácilmente.

Además, nos guardamos la cantidad de links que entran y salen de cada nodo. El primer dato será útil para calcular la métrica IN-DEGREE, y el segundo dato será útil para saber cuánto valdra P_{ij}^t , dado que es $\frac{1}{n_j}$, donde n_j es la cantidad de nodos salientes del nodo j .

Interfaz La interfaz de Matriz provee las siguientes operaciones:¹

- `Matriz(ifstream& in)`: constructor de la matriz. Recibe un archivo abierto del cual parseará todos los datos que necesita, en el formato adecuado (SNAP).
- `vector<double> multiplicar(vector<double> x)`: El propósito de esta función es autoexplicativo. Devuelve el resultado del producto Ax , donde A es la matriz de la clase. El algoritmo que utilizamos es el standard para multiplicar por matrices representadas en

¹Cuando se escribe la aridad de la función la misma puede no coincidir con la notación usada en C++. Esto está bien pues lo único que se busca aquí es dar una orientación de lo que hace cada función y no código preciso.

CRS, y además, como dijimos anteriormente, dividimos cada entrada por la cantidad de nodos salientes del nodo que corresponda.

2.3.2. MatrizDep

El módulo MatrizDep es el que se usará para representar a las matrices de conectividad de ligas deportivas.

Representación interna Como la matriz de conectividad en este caso, a diferencia del anterior, no suele ser esparsa (de hecho, en el caso general podría aproximarse bastante al grafo completo), entonces no fue necesario utilizar ninguna estructura compleja, y utilizamos simplemente la clásica representación de vector de vectores.

Además fue necesario utilizar otra estructura para almacenar los puntajes de acuerdo a otros métodos de ranqueo (por ejemplo, el standard) de la liga deportiva correspondiente.

Interfaz La interfaz de MatrizDep provee las siguientes operaciones:

- `MatrizDep(ifstream& in)`: constructor de la matriz. Recibe un archivo abierto del cual parseará todos los datos que necesita, en el formato adecuado (el explicitado en el tp).
- `vector<double> multiplicar(vector<double> x)`: autoexplicativa. Devuelve el resultado del producto Ax , donde A es la matriz de la clase. El algoritmo que utilizamos es el común y corriente, el mismo que se utiliza al hacer cuentas a mano con matrices.

2.3.3. Problema

Problema es un módulo que engloba todo lo relacionado al modelado y a la resolución de cada caso particular.

Adicionalmente, tenemos algunas funciones auxiliares:

Interfaz La interfaz de Problema provee funciones utilizadas tanto para el modelado de rankings de páginas web como para el modelado de rankings en ligas deportivas.

Las siguientes operaciones en lo que concierne a la resolución de problemas relacionados con el modelado de páginas web:²

- `vector<double> pagerank(Matriz p_trans, double c, double tolerancia)` : Se ocupa de aplicar el algoritmo de PageRank sobre la matriz `p_trans`, utilizando el algoritmo descrito por [Kam+03, Algoritmo 1]. Además, se testea luego de cada iteración si la diferencia (en norma 1³) es menor que la tolerancia. En ese caso, se termina de iterar y se considera que el vector resultante de la iteración actual es el resultado.

²Cuando se escribe la aridad de la función la misma puede no coincidir con la notación usada en C++. Esto está bien pues lo único que se busca aquí es dar una orientación de lo que hace cada función y no código preciso.

³ $\|x\|_1 = \sum_i |x_i|$

- **vector<uint>** indeg(**Matriz** p_trans): Devuelve el resultado de aplicarle el método de ranqueo indegree a la red del problema.

A continuación siguen los métodos utilizados para el modelado de rankings de ligas deportivas

- **vector<double>** metodopot(**MatrizDep** p, **double** tolerancia): Método de la potencia *vanilla*. Es el que se utilizará para resolver el problema de las ligas deportivas. Obtiene una sucesión de vectores, y cuando su diferencia de norma 1 es menor que la tolerancia pasada como parámetro se termina de iterar.

2.4. Experimentación

3. Resultados y discusión

3.1. PageRank y páginas web

3.2. PageRank y ligas deportivas

4. Conclusiones

5. Apéndices

5.1. Proposiciones

5.1.1. Proposición 1

Si $P \in R^{n \times n}$ es una matriz de transición, es decir $0 \leq P_{ij} \leq 1$, y $\sum_j P_{ij} = 1 \ \forall i \in \{1, \dots, n\}$, entonces el mayor autovalor en módulo de P^t es 1 o -1.

Demostración Primero veamos que si λ es autovalor de P^t , entonces $|\lambda| \leq 1$.

Vale que $\rho(P^t) \leq \|P^t\|$, $\rho(P^t)$ el radio espectral y $\|-\|$ cualquier norma inducida. En particular, si tomamos la norma 1, $\|P^t\|_1 = 1$, pues todas las columnas suman 1, pues P es de transición. Entonces $|\lambda| \leq \rho(P^t) \leq 1$.

Ahora, como las filas de P suman 1, si multiplico $P(1, \dots, 1)^t = (1, \dots, 1)^t$. Entonces, como P y P^t tienen los mismos autovalores, listo.

Referencias

- [Bar+] Richard Barrett y col. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. URL: <http://www.netlib.org/templates/templates.pdf>.
- [BF11] R. Burden y D. Faires. *Numerical Analysis*. Brooks/Cole, 2011.
- [BL06] Kurt Bryan y Tanya Leise. “The Linear Algebra behind Google”. En: *SIAM Review* 48.3 (2006), págs. 569-581.
- [BP98] Sergey Brin y Lawrence Page. “The anatomy of a large-scale hypertextual Web search engine”. En: *Computer Networks and ISDN Systems* 30.1-7 (abr. de 1998), págs. 107-117. ISSN: 01697552. DOI: [10.1016/S0169-7552\(98\)00110-X](https://doi.org/10.1016/S0169-7552(98)00110-X). URL: <http://linkinghub.elsevier.com/retrieve/pii/S016975529800110X>.
- [Data] <http://www.cs.toronto.edu/~tsap/experiments/datasets/>.
- [Datb] *DataHub*. <http://datahub.io>.
- [DB74] G. Dahlquist y A. Bjork. *Numerical Methods*. Prentice-Hall, 1974.
- [GMA08] Angela Y. Govan, Carl D. Meyer y Rusell Albright. “Generalizing Google’s PageRank to Rank National Football League Teams”. En: *Proceedings of SAS Global Forum 2008*. 2008.
- [GVL96] G. Golub y C. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.
- [Kam+03] Sepandar D. Kamvar y col. “Extrapolation methods for accelerating PageRank computations”. En: *Proceedings of the 12th international conference on World Wide Web*. WWW ’03. ACM, 2003, págs. 261-270. ISBN: 1-58113-680-3. DOI: [10.1145/775152.775190](https://doi.org/10.1145/775152.775190). URL: <http://doi.acm.org/10.1145/775152.775190>.
- [Kle99] Jon M. Kleinberg. “Authoritative Sources in a Hyperlinked Environment”. En: *J. ACM* 46.5 (sep. de 1999), págs. 604-632. ISSN: 0004-5411. DOI: [10.1145/324133.324140](https://doi.org/10.1145/324133.324140). URL: <http://doi.acm.org/10.1145/324133.324140>.
- [Sna] *Stanford Large Network Dataset Collection*. <http://snap.stanford.edu/data/#web>.