DEPARTAMENTO DE COMPUTACION

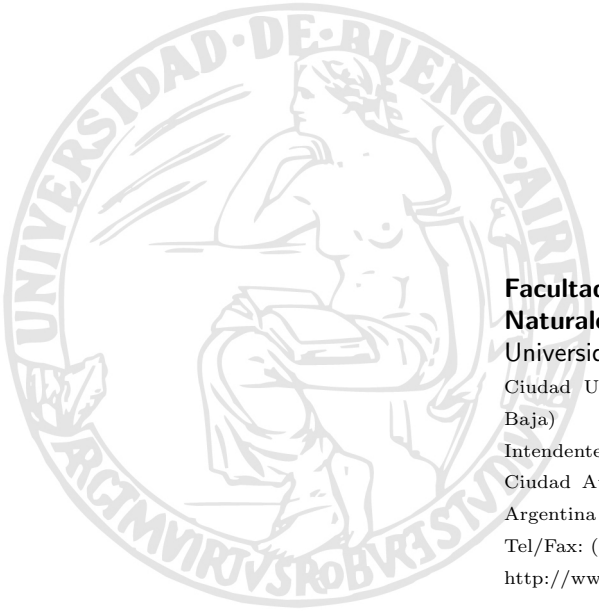
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1 - Page Rank

15 de abril de 2018 — Métodos Numéricos

Grupo "Nombre Del Grupo"

Integrante	LU	Correo electrónico
Facundo, Araujo	321/15	facalj_velez@hotmail.com
Cristian, Kubrak	456/15	Kubrakcristian@gmail.com
Marcela Alejandra, Herrera	1162/84	marcelaalejandraherrera@yahoo.com.ar
Luis Fernando, Greco	150/15	luifergreco@gmail.com



Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

Índice

1. Introducción

Modelado del problema

Para modelar la implementiación, utilizamos el *modelo de navegante aleatorio*[1]. Dado una probabilidad p buscamos resolver el sistema:

$$Ax = x \tag{1}$$

donde

$$A = a_{ij} = \begin{cases} (1-p)/n + (pw_i)/c & \text{si } c_j \neq 0 \\ 1/n & \text{si } c_j = 0 \end{cases} \tag{2}$$

Como se puede ver en (citar demo), la matriz A puede rescribirse como

$$A = pWD + ez^t \tag{3}$$

donde

$$D = d_{ij} = \begin{cases} 1/c_{ij} & \text{si } c_j \neq 0 \\ 0 & \text{si } c_j = 0 \end{cases}$$

e es un vector columna de dimension n y z es un vector columna cuyos componentes son:

$$z_j = \begin{cases} (1-p)/n & \text{si } c_j \neq 0 \\ 1/n & \text{si } c_j = 0 \end{cases}$$

De esta manera, la equación 1 se puede rescribir como

$$(I - pWD)x = \gamma e^t \tag{4}$$

donde $\gamma = z^t x$ es el factor de escala. Para el análisis de este método supusimos $\gamma = 1$

Introducción Teórica

El problema que se nos plantea es el de implementar un Page Rank, es decir, un método para ordernar páginas de acceso público de forma sistematizada y eficiente. Para esto tendremos que definir qué parametros vamos a tener en cuenta al momento de armar nuestro orden de páginas. Estos van a ser la cantidad de links *entrantes* a una página y la *calidad* de cada uno de estos, es decir, qué tan relevante es la página de la que proviene ese link.

Dado el volmunen de paginas que existen actualmente en Internet, resulta de vital importancia optimizar de alguna manera tanto la forma de almacenar como la forma de procesar la información. Para lograr dicha optimización decidimos utlizar matrices ralas (dónde muchos de sus elementos son ceros)

Para empezar a trabajar con matrices ralas nos vimos en la necesidad de buscar un método implementar este tipo de estructura, teniendo en cuenta un uso eficiente de los recursos del sistema (intentando sacar provecho del conocimiento que tenemos de entrada sobre las matrices). Es en este contexto que decidimos utilizar un híbrido entre el formato LL (List of Lists) y DOK (Dictionary of Keys).

Cuando comenzamos a pensar este problema, nos encontramos con que podíamos interpretarlo como una especie de Cadena de Markov. Esto es, un proceso estocástico en el que basados en nuestro estado actual, podemos describir la probabilidad futura -en nuestro caso el ranking, páginas ordenadas por probabilidad- sin importarnos los estados previos.

Luego podríamos interprear la matriz

$$A = pWD + ez^t \tag{5}$$

como que dada una página k en la cual estamos situados, en el próximo paso estaremos en alguna de las páginas a las que apunta k con probabilidad p ó bien en alguna página elegída al azar entre todas las conocidas, con probabilidad $(1-p)$.

La idea detrás del p es modelar la forma en la que se va a comportar nuestro "caminante aleatorio", es decir, la cantidad de veces que va a seguir alguno de los links salientes de las páginas en las que esté situado antes de saltar a una página totalmente aleatoria. Podemos notar que la cantidad de pasos antes de saltar a una página aleatoria es un número es una variable aleatoria con distribución $X \sim G_0\ (1-p)$.

Entonces, observamos que otra forma en la podríamos armar nuestro rankeador sería multiplicando la matriz de transición por si misma una cantidad de veces finita lo suficientemente grande como para luego poder utilizar con cierta confianza la Ley de los Grandes Números para Markov.

2. Desarrollo

Las relaciones entre páginas forman un grafo y resulta conveniente almacenarlos como una matriz de uno y ceros, done el un uno en la poscición ij representa que la página j apunta a la i. Aún así esta implementaciíon tiene un problema: debe almacenar n^2 elementos, donde solo importan m (la cantidad de unos). En la práctica, $m \ll n^2$ (ya que no todas las páginas se relacionan con todos). Es por esto que decidimos utilizar matrices *ralas*: un tipo de matriz donde solo importa donde hay unos, es decir, qué página apunta a cuál.

Como primer acercamiento, evaluamos utilizar dos conocidos conocidos métodos para el almacenamiento de matrices ralas: CSR (Compressed Sparse Row) y CSC (Compressed Sparse Column). En el primer caso nos encotramos con el problema de la dificultad de acceder a las colmunas, mientras que en el segundo, las filas. consideramos de vital importancia poder acceder tanto a filas como columnas en un tiempo razonable para operaciones tales como la multiplicación o la Eliminación Gausseana.

Luego de evaluar los requerimientos, tanto de complejidad como de espacio utilizado, optamos por implementar una estructura híbrida entre un DOK (Dictionary Of Keys) y una lista de listas. Utilizamos una estructura que consiste en un diccionario de diccionarios: en el primero almacenamos todas las filas y en el segundo sus elementos.

Respecto a la implementación, nos encontramos frente a la decisión de utilizar un diccionario ordenado implementado sobre una estructura autobalanceada (`std::map`) o un diccionario desordenado implementado sobre una tabla de hash (`std::unordered_map`). Si bien este úlitmo permite el acceso en $O(1)$ en promedio frente al acceso en $O(\log n)$ del diccionario ordenado, no resulta fácil iterar eficientemete por lo que terminamos decidiéndonos por la versión autobalanceada.

Nuestro objectivo es resolver el listem a 1 o equivalentemente el sistema 4, para lo cual resulta práctico y eficiente utilizar Eliminación Gausseana:

ELIMINACIÓN GAUSSEANA(A)

```
1  for k = 1 to n − 1
2      for i = k + 1 to n
3          for j = k + 1 to n
4              a_ij = a_ij − a_ik a_kj
```

Como la matriz que utlizamos para almacenar la informción es rala (no todos los elementos se encuentran definidos) por lo que además hay que asegurarse que los elementos estén definidos antes de operar con ellos. Más aún, dado que la resolución del sistema no es más que una aproximación, es necesario definir un $\varepsilon > 0$ para determinar qué tan bien queremos que aproxime a la solución real; si bien un ε más chico produciría una mejor aproximación, también ralentiza la ejecución del algoritmo. Es por esto, que junto con la esparsidad ($\delta = 1 - \frac{m}{n^2}$), decidimos hacer un análisis sobre el tamanño del ε por lo que el algoritmo de resolución queda:

```
ELIMINACIÓN GAUSSEANA RALA(A)
1  for k = 1 to n − 1
2      for i = k + 1 to n
3          for j = k + 1 to n
4              if a_ik.definido() and a_kj.definido()
5                  mult = a_ik/a_kk
6                  if a_ij.definido()
7                      a_ij = a_ij − mult ∗ a_kj
8                  else
9                      a_ij = −mult ∗ a_kj
```

Observar que en ningún momento verificamos que a_{kk} esté definido ya que la matriz permite hacer Eliminación Gausseana sin necesidad de pivoteo. (ver apéndice) Luego para determinar las soluciones del sistema[2]:

$$x_n = \frac{a_{n,n-1}}{a_{nn}}$$

$$x_i = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}x}{a_{ii}}$$

Por úlitmo, normalizamos el vector x de manera que $\sum_{i=1}^{n} |x_i| = 1$

Experimentación

En cuanto a la experimentación, pensamos en probar nuestra implementación de dos formas diferentes: de manera cualitativa y cuantitiativa. Para realizar el análisis cualitativo, generamos matrices aleatorias para luego variar los distintos parametros tales como probabilidad, tamaño de la matriz, cantidad de links y epsilon, para poder luego evaluar su incidencia. Para el análisis cualitativo, construimos casos particulares que nos resultaron interesantes analizar por diversos motivos.

Con respepcto al análisis cuantitiativo, varianmos los tres parámetros que considerados que podrían ser interesantes de analizar: el tamaño de la matriz n, la cantidad de links m, la probabilidad p y el ε.
En el primero de estos tests, creamos matrices de entrada aleatorias con tamaño n, probabilidad p y ε fijos, mientras variaban la cantidad de links m de la matriz generada, es decir cambiando qué tan rala es. A partir de esto, buscamos encontrar una relación entre la *sparsity* y el tiempo.
En este test nuestra expectativa fue encontrarnos con un incremento en el tiempo a medida que la *sparsity* de nuestra matriz de entrada era menor, considerando que la implementación fue diseñada para funcionar de manera eficiente en matrices de este tipo.

Como segundo test, buscamos evaluar incidencia del parametro p en el tiempo de ejecución en una matriz de tamaño m, cantidad de links n y ε fijos.
Al ver la ecuación 4 observamos que a media que el parametro p decrece, también lo hace el resultado de pWD y, por la forma en que implementamos la igualdad, tener elementos más pequeños en una matriz implica que más elementos de la misma podrían ser considerados ceros (dependiendo del ε que estemos utilizando).
Esto nos hace pensar que resolver la ecuación anterior en con un p relativamente

pequeño requeriría menos tiempo. Por último, no quisimos pasar por alto otro aspecto que consideramos que consideramos que podría tener cierta relevancia en este test, el número de condición. Como está señalado en el apéndice, vimos que a menor p, mejor condicionada está la matriz, entonces a medida que aumenta p, aumentará también la cota superior que tenemos para el número de condición, con lo que nuestro calculo podría volverse menos estable, en el sentido de a pesar de no conocer el error, nos exponemos a que sea más grande. Si bien esto nos abre el camino a nuevas experimentaciones, entendemos que el eje de este trabajo no es este, es por esto que simplemente lo mencionamos como un posible tema de interés para una próxima investigación.

Como última pruba cualitativa, mantuvimos fijos el tamaño de la matriz n, la cantidad de links m, la probabilidad p y variamos la precisión del ε. A la hora de realizar este test, estimamos que la variación del ε no afectará de manera significativa el tiempo de ejecución.

3. Resultados

Variación de la *sparcity* Al variar la relación de cantidad de links frente al tamaño de la matriz, estamos cambiando que tan rala es la matriz. Para analizar esto, tomamos en cuenta la *sparcity* la cual representaremos de ahora en más como $\delta = 1 - \frac{n}{m^2}$. A continuación se muestra el gráfico obtenido a partir de la experimentiación dejando fijos $n = 100$, $p = 0{,}8$, $\varepsilon = 10^{-5}$ y variando m desde 100 hasta 8000 de a 100.

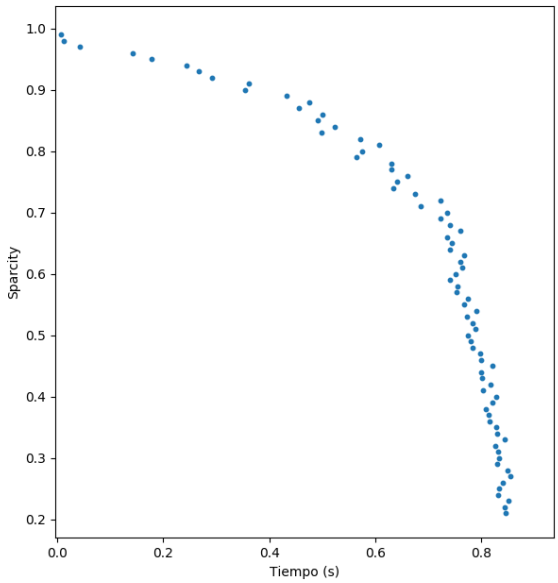


Figura 1: Gráfico de sparcity en función del tiempo

A partir de este gráfico podemos observar como, tal como fue predicho, que a medida que la matriz es menos rala ($\delta \approx 0$) aumenta considerablemente el tiempo que demora en ejecutarse el algoritmo.

Variación de la probabilidad Para analizar la incidencia de la probabilidad en el tiempo requerido, tomamos como parámetros fijos $n = 1000$, $m = 500$, $\varepsilon = 10^{-5}$ y variamos p desde 0,01 hasta 0,99 con un incremeto de 0,01 en cada paso.

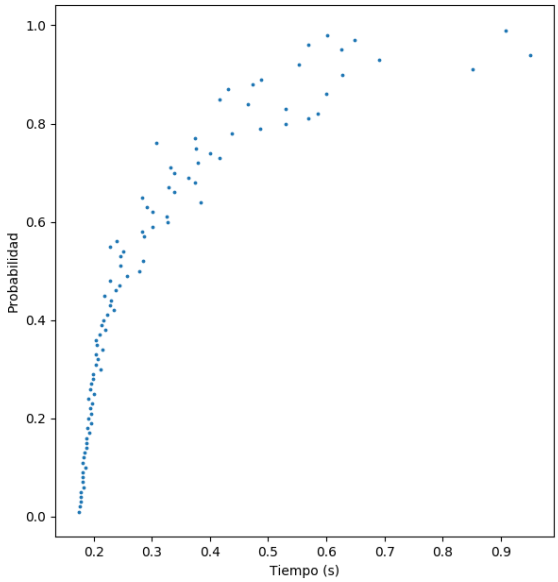


Figura 2: Gráfico de probabilidad en función del tiempo

Se puede apreciar como la elección de la probabilidad también afecta en el tiempo de ejecucón, tal como nos imáginabamos previo al experimento.

Variación del epsilon Para analizar cómo afecta el ε con el tiempo, tomamos como parámetros fijos $n = 1000$, $m = 500$, $\varepsilon = 10^{-5}$ y variamos p desde 0,01 hasta 0,99 con un incremeto de 0,01 en cada paso.

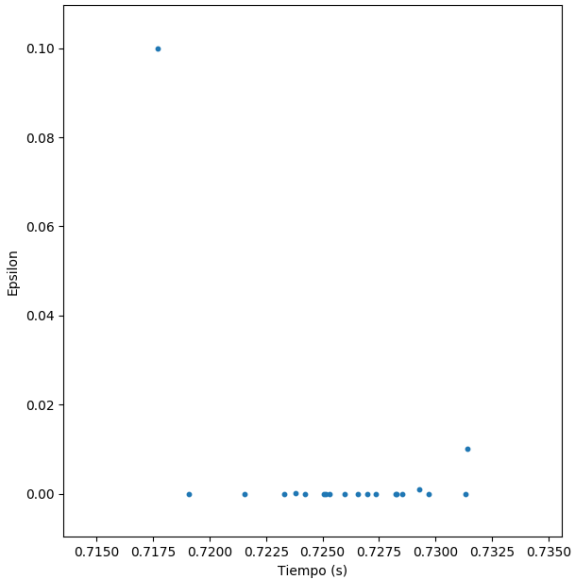


Figura 3: Gráfico de epsilon en función del tiempo

En este caso, pese a correr varias veces el test con distintos valores, no logramos apreciar correlación alguna entre la variación del ε y el tiempo.

4. Discusión

Discusión

5. Conclusiones

Conclusiones

6. Apendices

Demostraciones

1) Justificar que:

$$A = pWD + ez^t$$

Estudiemos como son los elementos de la matriz $A = pWD + ez^t$:

$$(pWD + ez^t)_{ij} = p(W\,D)_{ij} + (ez^t)_{ij} = p(\sum_{k=1}^{n} w_{ik}d_{kj} + 1\,z_j^t)$$

Como D es una matriz diagonal, los elementos distintos de cero únicamente pueden estar en la diagonal, con lo cual al hacer el producto se anulan todos los términos con $k \neq j$ y también aquellos términos donde $w_{ik} = 0$ lo cual ocurre cuando $c_j = 0$.

En consecuencia se puede reescribir:

$$(pWD + ez^t)_{ij} = p(w_{ij}d_{jj} + z_j^t)$$

Vamos a separar en casos para analizar los posibles valores que puede tomar el elemento (notar que por la definición de D, la condición $d_{jj} = 0$ es equivalente a $c_j = 0$):

caso 1: $w_{ij} = 0, d_{jj} = 0$ $(c_j = 0) \Rightarrow (pWD + ez^t)_{ij} = z_j^t = 1/n$

caso 2: $w_{ij} = 0, d_{jj} \neq 0$ $(c_j \neq 0) \Rightarrow (pWD + ez^t)_{ij} = z_j^t = (1-p)/n$

caso 3: $w_{ij} = 1, d_{jj} = 0$ $(c_j = 0) \Rightarrow (pWD + ez^t)_{ij} = z_j^t = 1/n$

caso 4: $w_{ij} = 1, d_{jj} \neq 0$ $(c_j \neq 0) \Rightarrow (pWD + ez^t)_{ij} = p/c_j + z_j^t = p/c_j + (1-p)/n$

Observando los casos 2 y 4 podemos observar que se pueden unificar en una condición equivalente: $(pWD + ez^t)_{ij} = w_{ij}p/c_j + z_j^t = p/c_j + (1-p)/n$

Uniendo todo lo anterior tenemos que:

$$(pWD + ez^t)_{ij} = \begin{cases} p\,w_{ij}/c_j + z_j^t = (1-p)/n + p/c_j & \text{si } c_j \neq 0 \\ 1/n & \text{si } c_j = 0 \end{cases} \tag{6}$$

Pero precisamente esto coincide con la definición de los elementos de la matriz A. Luego $A = pWD + ez^t$.

2) ¿Cómo se garantiza la aplicabilidad de la Eliminación Gaussiana? ¿La matriz $I - pWD$ está bien condicionada?¿Cómo influye el valor de p?

Primero veamos como se garantiza la aplicabilidad de Eliminación Gaussiana:

Sea $B = W\,D$,

$$B_{ij} = \sum_{k=1}^{n} w_{ik}d_{kj}$$

Por la definición de D, cuando $k \neq j$, $d_{kj} = 0$, por lo tanto se anulan los correspondientes términos de la sumatoria, quedando:

$$B_{ij} = w_{ij}d_{jj}$$

También sabemos por la definición de W que $w_{jj} = 0$ (porque no se consideran los autolinks), así que podemos deducir que los elementos de la diagonal van a ser iguales a cero, luego:

$$B_{ij} = \begin{cases} 0 & \text{si } i = j \\ 1/c_j & \text{si } i \neq j, w_{ij} \neq 0 \end{cases} \quad (7)$$

Y por lo tanto:

$$(I - pB)_{ij} = \begin{cases} 1 & \text{si } i = j \\ -p/c_j & \text{si } i \neq j \end{cases} \quad (8)$$

La matriz B tiene la particularidad de que la suma de los elementos de cada columna es igual a 1, ya que en cada columna hay c_j elementos de valor $1/c_j$. Por lo tanto en $I - pB$ la suma de los elementos de cada columna es $1 + c_j\, p/c_j$ con $|c_j\, p/c_j| < 1$ porque $0 < p < 1$. De esto se desprende que $(I - pB)^t$ es estrictamente diagonal dominante y como se probó en la teórica esto implica que sea no singular. Además, si una matriz es e.d.d. cada una de sus submatrices principales es e.d.d. y entonces todas ellas son no singulares. Por otra parte se puede probar que si una matriz tiene inversa, su traspuesta tiene inversa (ejercicio 23 c), Práctica 1). Luego podemos afirmar que $(I - pB)$ es no singular y que todas sus submatrices principales también son no singulares, esto garantiza que $(I - pB)$ tiene descomposición LU que es equivalente a decir que puede realizarse la Eliminación Gaussiana sin necesidad de pivoteo en ninguno de los pasos del proceso.

Con respecto al condicionamiento de la matriz, se puede aplicar una propiedad demostrada en la práctica (ejercicio 21, Práctica 2) la cual afirma que: Dada $M \in \mathbb{R}^{nxn}$ tal que $||M|| < 1$, I la matriz identidad de $\mathbb{R}^{nxn}$ i $||.||$ norma inducida por una norma vectorial: $I+M$ es inversible y $||(I+R)^{-1}|| \leq 1/(1-||M||)$.

De lo que vimos arriba acerca del formato de las matrices tratadas en el TP, también deducimos que tomando $||.||_1$, $M = -pWD$ estamos en las hipótesis del ejercicio, porque:

$$|| - pWD||_1 = ||pWD||_1 = |p|||WD||_1 = p * c_j * 1/c_j = p.$$

Y como $p < 1$, implica que:

$$||(I - pWD)^{-1}||_1 \leq 1/(1 - ||I - pWD||_1)$$

Si calculamos el número de condición,

$$K = ||I-pWD||||(I-pWD)^{-1}|| \leq (1+p)(1/(1-||pWD||) = (1+p)(1/(1-p||WD||) = (1+p)/(1-p)$$

Cuanto menor sea p mas cerca de 1 estará el número de condición, lo cual es lógico porque la matriz se parecerá más a I que tiene la característica de tener sus columnas ortogonales. A medida que crece p va aumentando el valor de K tendiendo a infinito si p tiende a 1. en el gráfico **??** se puede ver la relación entre el número de condición y el valor de p. Con un ejemplo se puede ver que con un valor relativamente cercano a 1 (0.9) el número de condición sigue siendo aceptable, y recién con valores mayores se puede decir que aumenta rápidamente. Supongamos p = 0.9: $K \leq (1{,}9/0{,}1) = 19$.

Para p=0.99: $K \leq (1{,}99/0{,}01) = 199$.

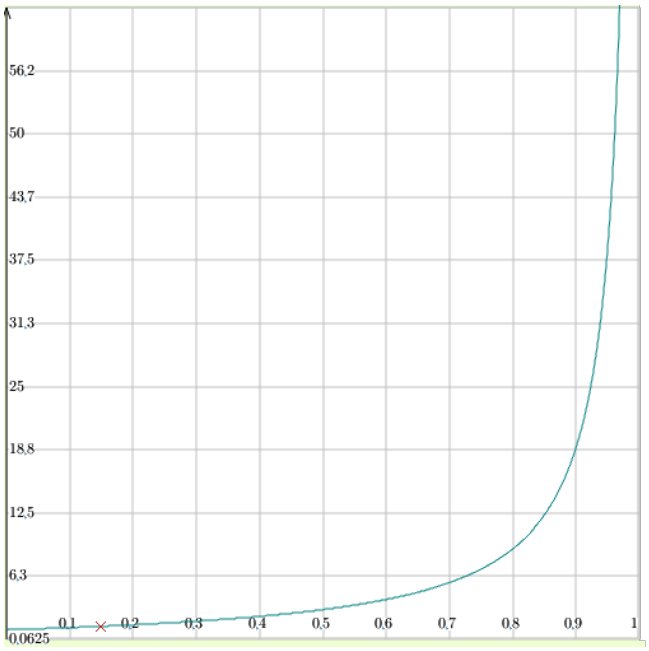


Figura 4: K en función de p

7. Referencias

Referencias

[1] Navegante aleatorio

[2] Richard L. Burden, J. Douglas Faires *Numerical Analysis*. (Ingles) [*Análisis numérico*]. International Thomson, :358-362, 9th Edition, 2010.