SAPIENZA
UNIVERSITÀ DI ROMA

Analisi dell'algoritmo di Google PageRank

Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea Triennale in Matematica

Iacopo Di Pippo
Matricola 1966046

Relatore
Prof. Gabriele Tolomei
Prof.ssa Elena Agliari

Anno Accademico 2022-2023

Analisi dell'algoritmo di Google PageRank
Sapienza Università di Roma

© 2023 Iacopo Di Pippo. Tutti i diritti riservati

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Versione: 02/05/2023

Email dell'autore: dipippo.1966046@studenti.uniroma1.it

Indice

Capitolo 1

Introduzione

Il World Wide Web è composto da quasi 50 miliardi di pagine Web. Da questa immensità di siti Web, gli utenti vorrebbero visitare un paio di siti su un determinato argomento. Ma come può un utente sapere quale fra queste miliardi di pagine Web dovrebbe visitare?

È qui che entrano in gioco i motori di ricerca. La maggior parte dei sistemi di ricerca utilizza un approccio a due fasi. Nella prima fase, un sistema tradizionale scansiona tutte le pagine Web contenenti le parole chiave inserite dall'utente durante la ricerca e assegna loro un punteggio basato sulla rilevanza. Date le dimensioni immense del World Wide Web, questa fase iniziale può produrre decine di migliaia di risultati. Nella seconda fase vengono ordinati i vari risultati in base alla loro importanza.

Google è il primo motore di ricerca che è riuscito a farlo in modo efficace. Il segreto di Google è il metodo PageRank. Questo metodo è stato sviluppato nel 1996 dai fondatori di Google, Larry Page e Sergey Brin. A ogni pagina del World Wide Web viene attribuito un punteggio di importanza (chiamato anche valore di PageRank). Ogni volta che un utente effettua una ricerca, Google ordina tutte le pagine Web che soddisfano la richiesta in base al loro valore di PageRank. L'utente riceverà i risultati della ricerca in questo ordine.

Il metodo PageRank di Google è un modo completamente oggettivo di valutare una pagina Web. Si basa su un modello matematico che utilizza solo la struttura ipertestuale del Web, ovvero i link che legano i vari siti. In questo elaborato discuteremo questo metodo. Vedremo che il vettore di PageRank (il vettore che contiene ogni valore di PageRank) è definito come un autovettore della cosiddetta matrice di Google $\mathcal{G}$.

Costruiremo la matrice di Google $\mathcal{G}$ in modo che sia stocastica e primitiva, permettendoci così di poter applicare il teorema di Perron-Froebenius per garantire l'esistenza e l'unicità del vettore di PageRank. Dimostreremo il teorema di Perron-Froebenius nel capitolo 4. Inoltre, nello stesso capitolo, dimostreremo un teorema che classifica gli autovalori della matrice di Google $\mathcal{G}$, utile per studiare la convergenza dei metodi nei capitoli successivi.

La matrice di Google, tuttavia, è estremamente grande: ha una dimensione di quasi 50 miliardi per 50 miliardi. Pertanto, il calcolo di un autovettore è tutt'altro che semplice. Nel capitolo 5 svilupperemo e discuteremo metodi numerici per approssimare questo autovettore il più velocemente possibile. Questi metodi numerici vengono poi messi alla prova, utilizzando più matrici di Google di piccole dimensioni, corrispondenti a più sottoinsiemi di pagine Web nel World Wide Web. I test e i risultati si trovano nel capitolo 6.

Vale la pena di notare che l'intera teoria di questo elaborato si basa sul documento originale di PageRank di Google di Brin e Page del 1998, vedi [**1**] nella bibliografia. È quasi certo che Google abbia ulteriormente sviluppato questo modello. L'attuale modello è tenuto segreto e pertanto la teoria e i risultati risultati potrebbero non essere del tutto aggiornati. Tuttavia, Google ha dichiarato che l'algoritmo attualmente utilizzato da Google si basa ancora sul modello originale di PageRank.

Capitolo 2

Conoscenze Preliminari

Definizione 2.0.1. *Definiamo* $\boldsymbol{e}$ *il vettore colonna con tutte le entrate pari a 1:*

$$\boldsymbol{e} = \begin{bmatrix} 1 \\ . \\ . \\ 1 \end{bmatrix}.$$

Definiamo inoltre I_1 *la matrice con tutte le entrate pari a 1, ovvero* $I_1 = \boldsymbol{e}\boldsymbol{e}^T$.

Definizione 2.0.2. *Siano* $\lambda_1, ..., \lambda_n$ *gli autovalori (reali o complessi) di una matrice* $A \in \mathbb{C}^{n\times n}$. *Definiamo il* ***raggio spettrale*** *della matrice* A *come:*

$$\rho(A) = max_{1\leq i\leq n}|\lambda_i|$$

Lemma 2.0.1. *Sia* A *una matrice* $A \in \mathbb{R}^{n\times n}$, *allora* $\lambda \in \mathbb{C}$ *è un autovalore di* A *se e solo se* λ *è un autovalore di* A^T.

Definizione 2.0.3. *Una matrice* $A \in \mathbb{R}^{n\times n}$ *si dice* ***substocastica*** *se valgono le seguenti proprietà:*

- $a_{ij} \geq 0$ *per ogni* $\forall i, j \in \{1, ..., n\}$
- $\sum_{j=1}^{n} a_{ij} \leq 1$ *per ogni* $1 \leq i \leq n$.

Definizione 2.0.4. *Una matrice* $A \in \mathbb{R}^{n\times n}$ *si dice* ***stocastica*** *se valgono le seguenti proprietà:*

- $a_{ij} \geq 0$ *per ogni* $\forall i, j \in \{1, ..., n\}$
- $\sum_{j=1}^{n} a_{ij} = 1$ *per ogni* $1 \leq i \leq n$.

Serie Geometrica per le matrici 2.0.1. *Sia $A \in \mathbb{R}^{n\times n}$ tale che $\rho(A) < 1$. Allora $\sum_{n=0}^{\infty} A^n$ converge, in particolare:*

$$(I_d - A)^{-1} = \sum_{n=0}^{\infty} A^n \tag{2.0.1}$$

Definizione 2.0.5. *Sia $A \in \mathbb{R}^{n\times n}$ una matrice substocastica. A è detta **irriducibile** se, per ogni coppia i, j, esiste un $n \in \mathbb{N}$ tale che $(A^n)_{ij} > 0$. Se questa proprietà non vale, la matrice viene detta riducibile.*

Definizione 2.0.6. *Sia $A \in \mathbb{R}^{n\times n}$, A è detta **aperiodica** se, per ogni $1 \leq i \leq n$, il periodo di i è uguale a 1, ovvero se il $MCD\{n \geq 1 : (A^n)_{ii} > 0\} = 1$. In particolare, se $A_{ii} > 0$ per ogni $1 \leq i \leq n$, allora A è aperiodica. Una matrice non aperiodica è detta periodica.*

Definizione 2.0.7. *Sia $A \in \mathbb{R}^{n\times n}$ una matrice substocastica. A è detta **primitiva** se è aperiodica e irriducibile.*

Definizione 2.0.8. *Un autovalore λ di una matrice A si dice **semplice** se è una radice semplice del polinomio caratteristico relativo a A o, equivalentemente, se λ è un autovalore con molteplicità algebrica 1.*

Definizione 2.0.9. *Sia $A \in \mathbb{R}^{n\times n}$ tale che tutti gli elementi della matrice A siano maggiori uguali a 0. Allora diciamo che A è una matrice **non negativa** e scriviamo $A \geq 0$.*

Algoritmo del metodo di Jacobi 2.0.1.
Dato il sistema lineare Ax=b:

$$\begin{cases} \text{Sia } x^{(0)} \text{ un vettore iniziale} \\ k = 0 \\ x^{(k+1)} = \mathbf{B}_j x^{(k)} + f \\ k = k + 1 \end{cases}$$

con $\mathbf{B}_j$ la matrice di iterazione del metodo di Jacobi: $\mathbf{B}_j = D^{-1}(E+F)$ e $f = D^{-1}b$ ($D = diag(diag(A)), E = -tril(A,-1), F = -triu(A,1)$).

Capitolo 3

Il metodo di PageRank

Il metodo PageRank è un algoritmo progettato dai fondatori di Google, Sergey Brin e Larry Page. Questo algoritmo costituisce la causa principale per cui Google è diventato il motore di ricerca più diffuso. A ogni pagina Web viene assegnato un punteggio che segna una scala di importanza. Maggiore è il punteggio ottenuto, più la pagina è importante e prima verrà trovata fra i risultati di ricerca.

L'idea alla base dell'algoritmo di PageRank è quella di sfruttare i link in uscita da ogni pagina Web, partendo dal presupposto che, se più pagine rimandano a un'altra in particolare, ha allora senso considerare quest'ultima pagina importante e darle un valore maggiore.

Il problema che Larry Page e Sergey Brin subito notarono è che il numero di link che mandano ad una pagina è facilmente manipolabile. Per limitare questa possibilità viene tenuto conto anche dell'importanza della pagina che contiene il link, dopotutto, se una pagina importante fa riferimento ad un'altra pagina, è chiaro che probabilmente anche quest'ultima sarà importante.

3.1. La formula di partenza del PageRank

Innanzitutto, è necessario dare qualche definizione preliminare per far scorrere più agevolmente il discorso. In questo caso ci adatteremo alla nomenclatura inglese, in quanto intuitiva e semplice. Data una pagina Web:

- gli **Outer link** sono i link in uscita della pagina.
- i **Back link** sono i link di altre pagine che rimandano alla pagina in questione, ovvero i link in entrata.

Brin e Page partirono da una semplice equazione per calcolare il punteggio di una pagina Web:

$$p_i = \sum_{j \in \mathcal{I}(i)} \frac{1}{|U(j)|} p_j \tag{3.1.1}$$

Dove con p_i si intende il valore assegnato alla pagina i, $\mathcal{I}(i)$ è l'insieme di pagine che rimandano alla pagina i e $|U(j)|$ è il numero di outer link della pagina j. Questa equazione è ben definita perché, nel caso $|U(j)| = 0$, allora $j \notin \mathcal{I}(i)$.

Il vettore $p \in \mathbb{R}^{n\times 1}$ che risolve l'equazione 3.1.1 è il vettore che corrisponde ai nostri scopi: il punteggio della pagine i, infatti, dipende dalla sommatoria sul numero di back link, più link mandano alla pagina i, più i sarà importante; dipende dal valore di p_j, più è importante la pagina j, più importante sarà la pagina a cui è collegato; e dipende da $|U(j)|$, più una pagina ha outer link, meno peso ognuno di questi link avrà.

Il problema di questa formula è che il valore della pagina Web i dipende da quelle delle pagine j, che non conosciamo. Per questo motivo la formula viene considerata in maniera ricorsiva:

$$\begin{cases} p_i^{k+1} = \sum\limits_{j\in\mathcal{I}(i)} \frac{1}{|U(j)|} p_j^k \\ p_i^0 = \frac{1}{n} \qquad \forall i \in \{1, ..., n\} \end{cases} \tag{3.1.2}$$

Con p_i^k si intende il punteggio della pagina i alla k-esima iterazione. Il processo è inizializzato con $p_i^0 = \frac{1}{n}\ \forall i \in \{1, ..., n\}$, con n il numero totale di pagine Web considerate.

Ponendo l'equazione in forma ricorsiva otteniamo un algoritmo. L'obiettivo è fare sì che l'algoritmo converga a un vettore $p \in \mathbb{R}^{n\times 1}$ che risolva l'equazione 3.1.1. Come vedremo nel terzo paragrafo questo algoritmo incorre in vari problemi di convergenza. La soluzione che Brin e Page hanno trovato è quella di modificare l'algoritmo in modo da poter usare il teorema di Perron-Frobenius per assicurarsi l'esistenza e l'unicità della soluzione di 3.1.1.

3.2. La formula di partenza in forma matriciale

L'equazione 3.1.2 calcola il valore di una pagina Web per volta. Si può semplificare di molto la notazione dell'equazione definendo il vettore riga $\boldsymbol{p} \in \mathbb{R}^{n\times 1}$ e la matrice $\mathcal{H} \in \mathbb{R}^{n\times n}$, in modo da eliminare la sommatoria e rendere l'equazione 3.1.2 un prodotto matriciale.

Definizione 3.2.1.
Denotiamo $\boldsymbol{p}$ il vettore riga $n \times 1$ che all'entrata i-esima assegna il valore della i-esima pagina Web. Denotiamo inoltre $\boldsymbol{p}^k$ il vettore riga $n \times 1$ che all'entrata i-esima assegna il valore dell'i-esima pagina Web alla k-esima iterazione dell'algoritmo di PageRank.

Definizione 3.2.2.
Sia $\mathcal{H} \in \mathbb{R}^{n\times n}$ tale che $\forall i, j \in \{1, ..., n\}$

$$\mathcal{H}_{ij} = \begin{cases} 1/|U(i)| & \text{se esiste un link che parta dalla pagina i e finisce nella j} \\ 0 & \text{altrimenti} \end{cases}$$

Denotiamo questa matrice come la ***matrice dei link****.*

Osserviamo che la matrice dei link può essere definita partendo da un grafo orientato, considerando i nodi come le pagine Web e gli archi come i link.

L'equazione 3.1.2 diventa quindi:

$$\boldsymbol{p}^{(k+1)} = \mathcal{H}^T \boldsymbol{p}^{(k)} \tag{3.2.1}$$

Facciamo subito delle osservazioni riguardo l'equazione 3.2.1:

- Ogni iterazione dell'equazione 3.2.1 ha un costo di computazione di $\mathcal{O}(n^2)$.
- $\mathcal{H}$ è una matrice sub-stocastica, infatti per costruzione ogni elemento è non negativo e la somma di ogni riga è al massimo 1.
- $\mathcal{H}$ è una matrice molto sparsa, poiché la maggior parte degli elementi è 0. Ciò segue dal fatto che il Web è costituito da miliardi di pagine Web ma ogni pagina Web ha pochissimi outer link.

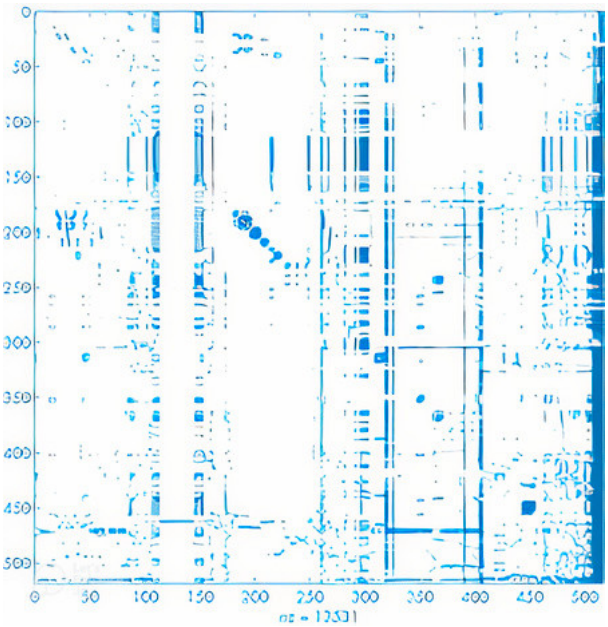

Figura 3.1. Esempio di matrice sparsa. Questo grafico rappresenta la matrice dei link di un sottoinsieme del Web composto da tutte le pagine Web i cui link iniziano per www.mathworks.com. La colonna più densa a destra rappresenta quei link che sono comuni a più pagine.

Computazionalmente avere una matrice molto sparsa è ottimo, non solo perché occupa molto meno spazio di memoria, ma anche perché un'iterazione dell'equazione 3.2.1 ha come costo di computazione $\mathcal{O}(nnz(\mathcal{H}))$, con $nnz(\mathcal{H})$ il numero di elementi non nulli della matrice $\mathcal{H}$. In media ogni pagina Web ha circa 10 outer link, questo implica che il numero di elementi non nulli nella matrice in media sarà $10n$ e quindi il costo di computazione passa da $\mathcal{O}(n^2)$ a $\mathcal{O}(n)$.

- Infine la matrice $\mathcal{H}$ è una matrice periodica e riducibile, questo perché il Web consiste di moltissime pagine Web senza alcun link né in entrata né in uscita e spesso accade che due pagine Web siano collegate unicamente fra di loro, generando cicli.

E' doveroso fare un esempio per comprendere al meglio il funzionamento dell'equazione 3.2.1. Consideriamo il caso di 6 pagine Web, collegate fra loro in questo modo:

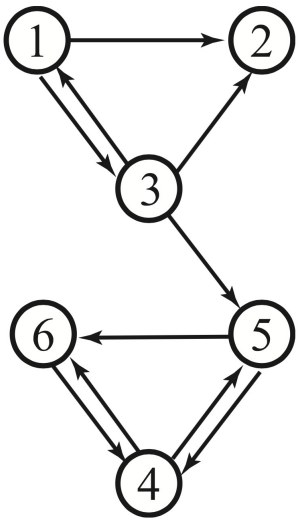

Figura 3.2. Grafo diretto che rappresenta i collegamenti fra le 6 pagine Web.

La matrice dei link $\mathcal{H}$ associata a questo grafo è la seguente:

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

Osservazione 3.2.1. *Osservando la tabella 3.1 si può capire il funzionamento dell'equazione 3.2.1. Ad ogni iterazione ogni nodo cede parte del suo punteggio ai nodi a cui è legato dai suoi outer link e, viceversa, guadagna valore prendendolo dai nodi a cui è legato dai suoi back link. Questo vuol dire che i nodi 1 e 3, i quali hanno tanti outer link e pochi back link, tenderanno a cedere a ogni iterazione parte del loro valore, ricevendone in cambio ben poco. Il contrario accade con i nodi 4, 5 e 6, i quali, ad ogni iterazione, prendono e si distribuiscono il valore che ottengono dai nodi 1, 2 e 3, non condividendo mai il loro punteggio con gli altri 3 nodi.*

Applicando ora l'algoritmo 3.2.1, inizializzando il vettore $\boldsymbol{p}^0 = \frac{1}{n}\boldsymbol{e}$, otteniamo:

Tabella 3.1. Prime iterazioni dell'equazione 3.2.1 applicata al grafo della figura 3.2.

Iterazione 0	Iterazione 1	Iterazione 2	Iterazione 3	Classifica alla Iter. 3
$p_1^0 = 1/6$	$p_1^1 = 1/18$	$p_1^2 = 1/36$	$p_1^3 = 1/108$	6
$p_2^0 = 1/6$	$p_1^1 = 5/36$	$p_1^2 = 1/18$	$p_2^3 = 5/216$	4
$p_3^0 = 1/6$	$p_1^1 = 1/12$	$p_1^2 = 1/36$	$p_3^3 = 1/72$	5
$p_4^0 = 1/6$	$p_1^1 = 1/4$	$p_1^2 = 17/72$	$p_4^3 = 13/48$	1
$p_5^0 = 1/6$	$p_1^1 = 5/36$	$p_1^2 = 11/72$	$p_5^3 = 55/432$	3
$p_6^0 = 1/6$	$p_1^1 = 1/6$	$p_1^2 = 14/72$	$p_6^3 = 7/36$	2

3.3. I problemi del metodo iterativo

L'equazione iterativa 3.2.1 lascia però ancora moltissime questioni in sospeso:

- Questo processo iterativo converge?
- Sotto quali condizioni su $\mathcal{H}$ converge?
- Con che rapidità converge?
- La convergenza dipende dal vettore iniziale $\boldsymbol{p}^0$?
- Il problema convergerà a qualcosa di sensato rispetto quello che stavamo cercando?

Brin e Page inizialmente avevano usato come vettore iniziale $\boldsymbol{p}^0$ il vettore tale che $\boldsymbol{p}^0 = 1/n \quad \forall i \in \{1, ..., n\}$. Sono subiti accorsi in vari problemi usando l'equazione 3.2.1 con questo vettore iniziale.

Il primo problema lo hanno riscontrato con le cosiddette **fosse di valore**, ovvero quelle pagine Web che continuano ad accumulare valore a ogni iterazione, monopolizzando il valore senza condividerlo con altre pagine Web. Nel semplice esempio rappresentato in figura 3.3 il nodo numero 3 è una fossa di valore.

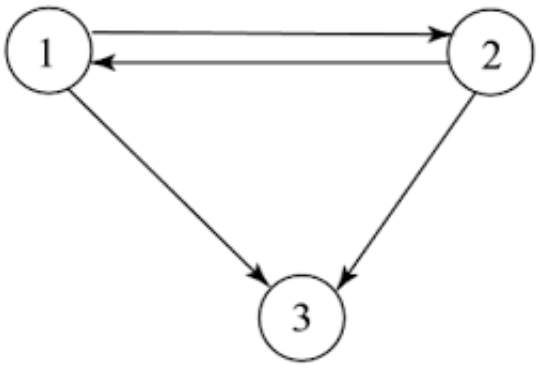

Figura 3.3. La più semplice rappresentazione di un grafo con una fossa di valore.

Un esempio più complicato di fossa di valore si può osservare nella figura 3.2, dove i nodi 4,5 e 6 si coalizzano per monopolizzare il valore delle pagine. Infatti, dopo

solamente 13 iterazioni dell'equazione 3.2.1, $p^{13} = (0 \quad 0 \quad 0 \quad 2/3 \quad 1/3 \quad 1/5)$. Il problema con queste configurazioni è che, mentre una parte dei nodi prende moltissimo valore, molte pagine rimangono senza. E' chiaro che classificare le pagine Web non è facile se la maggior parte dei nodi hanno come valore 0. In generale vorremmo che il vettore che descrive la classifica delle pagine Web sia positivo.

Un altro problema lo hanno riscontrato nel caso in cui le pagine Web formino un ciclo. Nel caso in figura 3.4 il nodo 1 punta solamente al nodo 2 e viceversa.

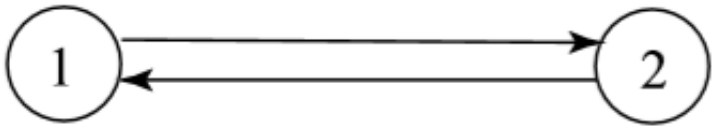

Figura 3.4. La più semplice rappresentazione di un grafo con un ciclo.

Supponiamo che la distribuzione iniziale sia $\boldsymbol{p}^0 = (1 \quad 0)$, in questo caso l'algoritmo definito dall'equazione 3.2.1 non convergerà mai. Infatti, a ogni iterazione, il vettore $\boldsymbol{p}^k$ varrà $(1 \quad 0)$ per k pari e $(0 \quad 1)$ per k dispari.

3.4. Il Web come Grafo orientato e il Navigatore random

Per risolvere questi problemi Brin e Page hanno deciso di sfruttare la teoria sulle Catene di Markov. Si sono accorti che, se avessero modificato la matrice dei link $\mathcal{H}$ in modo da renderla prima stocastica e poi primitiva, si sarebbero assicurati l'esistenza e l'unicità della soluzione grazie al teorema di Perron-Frobenius. Inoltre una matrice primitiva è anche aperiodica, eliminando così la presenza dei cicli.

E' interessante notare che Brin e Page non hanno mai usato, nelle loro pubblicazioni, notazioni relative alle Catene di Markov. Infatti, per giustificare gli aggiustamenti che hanno apportato alla matrice $\mathcal{H}$, sono ricorsi alla nozione di "navigatore random". Si sono immaginati un utente del Web ideale, il navigatore random, che, partendo da una pagina Web fissa, osserva gli outer link di quella pagina e visita con probabilità uniforme uno dei link in uscita. Infine, una volta visitata la nuova pagina Web, itera il procedimento, continuando a visitare pagine Web all'infinito seguendo la struttura ipertestuale del Web.

In questo modo la probabilità con cui il navigatore random passa da una pagina i a una pagina j è proprio descritto dall'entrata $\mathcal{H}_{ij}$ della matrice dei link. E' quindi chiaro il collegamento con le Matrici di Markov: ogni nodo è uno stato e la matrice di transizione nell'entrata ij descrive la probabilità con cui il navigatore random passa dalla pagina i alla pagina j.

Immaginandosi di far navigare questo utente nel Web all'infinito, la frazione di tempo che passa in una pagina Web costituisce il valore di quella pagina.

Tuttavia questo metodo incorre in un altro problema: il navigatore si blocca ogni volta che compare una pagina Web senza link in uscita. Per risolvere questo problema Brin e Page decidono di permettere al Navigatore Random di visitare con probabilità uniforme qualsiasi altra pagina Web esistente nel caso incontrasse una pagina senza link in uscita. Dobbiamo quindi aggiustare la matrice $\mathcal{H}$ in modo che sia coerente con le nuove possibilità di movimento del navigatore random. Per questo motivo definiamo una nuova matrice:

Definizione 3.4.1.
Sia $\mathcal{S}$ la matrice corrispondente ai movimenti del nuovo navigatore random, allora $\mathcal{S} = \mathcal{H} + \frac{1}{n} a \boldsymbol{e}^T$, dove con $a \in \mathbb{R}^{n\times 1}$ si indica il vettore tale che

$$a_i = \begin{cases} 1 & \textit{se la pagina } i \textit{ non ha link in uscita} \\ 0 & \textit{altrimenti} \end{cases}$$

Denotiamo questa matrice come la ***matrice del navigatore random****.*

Osservazione 3.4.1. *La matrice del navigatore random $\mathcal{S}$ è per costruzione una matrice stocastica, quindi può essere la matrice di transizione di una catena di Markov. L'unico problema è che questa matrice non è necessariamente primitiva. L'obiettivo di Page e Brin è proprio quello di rendere la matrice primitiva, assicurandosi così l'esistenza e l'unicità della soluzione.*

Osservazione 3.4.2. *Costruendo in questo modo la matrice del navigatore random abbiamo perso una caratteristica molto importante dal punto di vista computazionale, ovvero il fatto che fosse sparsa. Tuttavia, dal momento che $\mathcal{S}$ è combinazione lineare della matrice $\mathcal{H}$, questo non causerà molti problemi dal punto di vista computazionale.*

Riprendendo l'esempio basato sulla figura 3.2, calcoliamo la matrice del navigatore random partendo dalla matrice dei link, notando che $a = (0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)$, in quanto il nodo 2 è l'unico senza link in uscita. Otteniamo quindi:

$$\mathcal{S} = \mathcal{H} + \frac{1}{n} a \boldsymbol{e}^T = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{S} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

3.5. La matrice di Google

L'ultima modifica fatta da Brin e Page è stata quella di permettere al navigatore random di non sfruttare i collegamenti ipertestuali della pagina Web che visitava, dandogli la possibilità di teletrasportarsi in una pagina Web scelta casualmente. Per fare ciò introdussero il **parametro di teletrasporto $\boldsymbol{\alpha}$**, un parametro che descrive la probabilità che il navigatore random segua uno dei link in uscita.

Ogni volta che il navigatore random visita una pagina Web ha quindi una probabilità di $1-\boldsymbol{\alpha}$ di teletrasportarsi in una pagina a caso del Web, scelta uniformemente su tutte le pagine; nel restante dei casi il navigatore Random dovrà seguire i collegamenti ipertestuali e scegliere uniformemente uno fra gli outer link. Brin e Page giustificarono l'introduzione del parametro di teletrasporto come il parametro che descrive la probabilità con cui l'utente medio decide di inserire un altro link mentre naviga nel Web.

Il valore di $\boldsymbol{\alpha}$ decide quanto spesso il navigatore random segue i collegamenti ipertestuali. E' chiaro che più $\boldsymbol{\alpha}$ è vicino a 0 meno l'algoritmo di PageRank dipenderà dai collegamenti ipertestuali, portando così ad un valore finale più uniforme, mentre più $\boldsymbol{\alpha}$ sarà vicino a 1 più lentamente l'algoritmo convergerà. Brin e Page usarono come parametro di teletrasporto $\boldsymbol{\alpha} = 0.85$, nei prossimi capitoli discuteremo perché hanno scelto questo valore, ma per adesso lo prendiamo per dato.

Anche in questo caso c'è bisogno di aggiustare la matrice $\mathcal{S}$ in modo che sia coerente con le possibilità di movimento del navigatore random. Definiamo quindi una nuova matrice:

Definizione 3.5.1.
Sia $\mathcal{G} \in \mathbb{R}^{n\times n}$ la matrice $\mathcal{S}$ aggiustata per permettere il teletrasporto al navigatore random. Allora $\mathcal{G} = \boldsymbol{\alpha}\mathcal{S} + \frac{1}{n}(1-\boldsymbol{\alpha})\mathbb{I}_1$ con $I_1 \in \mathbb{R}^{n\times n}$ la matrice composta da 1 in tutte le entrata. Denotiamo questa matrice $\mathcal{G}$ come la ***matrice di Google****.*

Osservazione 3.5.1. *Facciamo delle osservazioni sulla matrice di Google:*

- *$\mathcal{G}$ è una matrice stocastica*
- *$\mathcal{G}$ è irriducibile*
- *$\mathcal{G}$ è aperiodica*
- *$\mathcal{G}$ è completamente densa. Fortunatamente ,come vedremo fra poco, $\mathcal{G}$ può essere scritta come combinazione lineare della matrice sparsa $\mathcal{H}$ e questo aiuterà moltissimo con la complessità computazionale.*

Brin e Page hanno quindi modificato il metodo di PageRank iniziale basato sull'equazione 3.2.1 modificando due volte la matrice dei link $\mathcal{H}$. L'equazione 3.2.1 diventa quindi:

$$\boldsymbol{\pi}^{(k+1)} = \mathcal{G}^T\boldsymbol{\pi}^{(k)} \tag{3.5.1}$$

che è semplicemente il metodo delle potenze applicato a $\mathcal{G}$.

Definizione 3.5.2.
Definiamo ora il ***vettore di PageRank*** *come l'unico vettore che soddisfa queste proprietà:*

1. $\boldsymbol{\pi} = \mathcal{G}^T \boldsymbol{\pi}$
2. $\sum_{i=1}^{n} \boldsymbol{\pi}_i = 1$
3. $\boldsymbol{\pi}_i > 0$ *per ogni* i *tale che* $1 \leq i \leq n$

L'importanza di una pagina Web non può essere misurata in maniera oggettiva. Ciò nonostante, seguendo il ragionamento del navigatore random, si trova un vettore di ordinamento, ovvero quello di PageRank, che si è rilevato essere molto efficace nel capire l'importanza delle pagine Web. Inoltre, poiché $\mathcal{G}$ è primitiva, possiamo finalmente applicare il teorema di Perron-Frobenius, garantendo così l'unicità e l'esistenza dell'autovettore della matrice $\mathcal{G}^T$ di autovalore 1, ovvero garantendo l'unicità e l'esistenza del vettore di PageRank.

Continuiamo l'esempio iniziato nel paragrafo 2 rispettivo alla figura 3.2. Possiamo ora calcolarci la matrice di Google di quel grafo, mettendo però come parametro di teletrasporto $\boldsymbol{\alpha} = 0.9$ per semplificare i calcoli:

$$\mathcal{G} = \boldsymbol{\alpha}\mathcal{S} + \frac{1}{n}(1-\boldsymbol{\alpha})\mathbb{I}_1 =$$

$$= \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} + \frac{(1-\alpha)}{n} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathcal{G} = \begin{bmatrix} 1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\ 1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\ 1/60 & 1/60 & 1/60 & 1 & 1/60 & 1/60 \end{bmatrix}$$

Ora possiamo calcolare il vettore di PageRank relativo alla matrice appena calcolata:

$$\boldsymbol{\pi}^T = (.03721 \quad .05396 \quad 0.04151 \quad .3751 \quad .206 \quad .2862)$$

Possiamo ora interpretare il risultato. Il fatto che $\boldsymbol{\pi}_1 = 0.03721$ vuol dire che il 3.721% delle volte il navigatore random visiterà il nodo 1. Dato il vettore di PageRank possiamo classificare le pagine Web della figura 3.2 in ordine decrescente rispetto l'importanza: (4 6 5 2 3 1). Come avevamo detto all'inizio, il nodo 1 e il nodo 3 sono i nodi meno importanti, mentre il nodo 4 è il più importante di tutti.

Nei successivi capitoli studieremo come calcolare in modo computazionalmente vantaggioso il vettore di PageRank, sfruttando il fatto che $\mathcal{G}$ può essere scritta come combinazione lineare della matrice $\mathcal{H}$, una matrice molto sparsa.

Descritto il metodo e l'algoritmo, non rimane che dimostrare due punti molto importanti:

1. Dimostrare che l'algoritmo iterativo dato dall'equazione 3.5.1 converge.
2. Mostrare con che velocità asintotica converge.

Capitolo 4

Risultati teorici

In questo capitolo verrà dimostrata l'unicità del vettore di PageRank sfruttando la molteplicità algebrica. In particolare verrà dimostrato che l'autovalore 1, corrispondente alla matrice $\mathcal{G}^T$, ha una molteplicità algebrica pari a uno. Questo implica che gli autovettori della matrice $\mathcal{G}^T$ rispetto l'autovalore 1 sono definiti a meno di uno scalare. Il vettore di PageRank non è altro che uno di questi autovettori e, poiché lo abbiamo definito in modo che il prodotto scalare con $\boldsymbol{e}$ sia 1, questo ci assicura l'unicità. Tale risultato sarà ottenuto usando una parte del teorema di Perron-Frobenius.

Inoltre faremo uso del teorema dei cerchi di Gershgorin per dimostrare vari lemmi. Questi lemmi saranno necessari per dimostrare un teorema che classifica e ordina gli autovalori della matrice di Google. Questa conoscenza sugli autovalori della matrice di Google sarà necessaria per capire la velocità asintotica di convergenza del metodo delle potenze.

Infatti, come vedremo, la velocità asintotica di convergenza del metodo dipenderà dal rapporto fra i due più grandi autovalori. Noteremo, inoltre, come questo rapporto dipenderà in maniera decisiva dalla scelta del parametro di teletrasporto $\boldsymbol{\alpha}$ fatta in precedenza.

4.1. Il teorema dei cerchi di Gershgorin

Teorema dei cerchi di Gershgorin.
Sia $\mathbf{A}$ *una matrice* $n \times n$ *di elementi reali o complessi. Allora ogni autovalore* λ *di* $\mathbf{A}$ *soddisfa la seguente proprietà:*

$$\lambda \in \bigcup_{i=1}^{n} \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}| \right\} \tag{4.1.1}$$

Quindi ogni autovalore di $\mathbf{A}$ *è contenuto nell'unione dei dischi centrati in* a_{ii} *e di raggio* $\sum_{j=1,j\neq i}^{n} |a_{ij}|$.

Dimostrazione. Siano λ e v rispettivamente autovalore e autovettore della matrice $\mathbf{A}$. Abbiamo quindi che $\mathbf{A}v = \lambda v$. Nella i-esima componente l'equazione diventa:

$$\sum_{j=1}^{n} a_{ij} v_j = \lambda v_i,$$

da cui

$$(a_{ii} - \lambda)\, v_i = -\sum_{j=1,j\neq i}^{n} a_{ij} v_j.$$

Prendendo i moduli di entrambi i membri e usando la disuguaglianza triangolare otteniamo:

$$|a_{ij} - \lambda| \cdot |v_i| \leq \sum_{j=1,j\neq i}^{n} |a_{ij}| \cdot |v_j|. \tag{4.1.2}$$

Sia h un indice tale che $|v_h| \geq |v_j|$ per $1 \leq j \leq n$. Allora applichiamo l'equazione 4.1.2 con $i = h$ e dividendo per $|v_h|$ otteniamo :

$$|a_{hh} - \lambda| \leq \sum_{j=1,j\neq h}^{n} |a_{hj}| \cdot \frac{|v_j|}{|v_h|} \leq \sum_{j=1,j\neq i}^{n} |a_{ij}|$$

dove l'ultima uguaglianza segue dal fatto che $\frac{|v_j|}{|v_h|} \leq 1$ per come abbiamo scelto h. Abbiamo quindi la tesi. □

Da questo importante teorema deriva immediatamente un lemma molto intuitivo utile nel caso di matrice stocastiche:

Lemma 4.1.1. *Sia* $\mathbf{A}$ *una matrice stocastica* $n \times n$. *Allora 1 è un autovalore di* $\mathbf{A}$. *Inoltre 1 è il più grande autovalore di* $\mathbf{A}$.

Dimostrazione. Sia $\boldsymbol{e}$ il vettore $n \times 1$ tale che $\boldsymbol{e} = (1, ..., 1)$. Allora

$$\mathbf{A}\boldsymbol{e} = \begin{pmatrix} \sum_{i=1}^{n} a_{1j} \cdot 1 \\ \sum_{i=1}^{n} a_{2j} \cdot 1 \\ \cdot \\ \cdot \\ \cdot \\ \sum_{i=1}^{n} a_{nj} \cdot 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{pmatrix} = \boldsymbol{e}$$

Quindi $\boldsymbol{e}$ è un autovettore di autovalore 1.

Ora applicando il teorema di Gershgorin su $\mathbf{A}$, otteniamo che per ogni autovalore z di $\mathbf{A}$ deve esistere una $1 \leq i \leq n$ tale che $|z - a_{ii}| \leq \sum_{j=1,j\neq i}^{n} |a_{ij}|$. Da questo deriva che:

$$|z| = |z - a_{ii} + a_{ii}| \leq |z - a_{ii}| + |a_{ii}| \leq \sum_{j=1,j\neq i}^{n} |a_{ij}| + |a_{ii}| = \sum_{j=1}^{n} |a_{ij}| = 1$$

Quindi 1 è il più grande autovalore di $\mathbf{A}$. □

4.2. Gli autovalori della matrice di Google $\mathcal{G}$

In questo paragrafo verranno mostrati i risultati ottenuti sugli autovalori della matrice di Google $\mathcal{G}$. Il teorema seguente li descrive, ma per dimostrarlo, è necessario dimostrare altri lemmi preliminari.

Teorema sugli autovalori della matrice di Google.
Sia $\mathcal{G}$ la matrice di Google come definita in 3.5.1 e siano $\lambda_1, \lambda_2, ..., \lambda_n$ gli autovalori della matrice $\mathcal{G}^T$ in ordine decrescente. Allora:

1. $\lambda_1 = 1$
2. $\lambda_2 \leq \boldsymbol{\alpha}$ *con* $\boldsymbol{\alpha}$ *il parametro di teletrasporto*

Lemma 4.2.1. *Per ogni $\lambda \in \mathbb{C}$ tale che $|\lambda| > \boldsymbol{\alpha}$, la matrice $\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T$ è invertibile.*

Dimostrazione. Useremo il teorema di Gershgorin per mostrare che 0 non è un autovalore di $M := \lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T$, questo implica che M sia invertibile.

Sia $1 \leq i \leq n$ un indice arbitrario. Allora

$$\sum_{j \neq i} |M_{ij}| = \boldsymbol{\alpha} \sum_{j \neq i} \mathcal{S}_{ij} = \boldsymbol{\alpha} \left(\sum_{j=1}^{n} \mathcal{S}_{ij} - \mathcal{S}_{ii} \right) = \boldsymbol{\alpha} \left(1 - \mathcal{S}_{ii}\right),$$

usando il fatto che $\mathcal{S}$ è una matrice stocastica. Quindi, per la disuguaglianza triangolare inversa,

$$|M_{ii}| = |\lambda - \boldsymbol{\alpha}\mathcal{S}_{ii}| \geq ||\lambda| - |\boldsymbol{\alpha}\mathcal{S}_{ii}|| \geq |\lambda| - |\boldsymbol{\alpha}\mathcal{S}_{ii}| > \boldsymbol{\alpha} - \boldsymbol{\alpha}\mathcal{S}_{ii}$$

Otteniamo quindi:

$$|M_{ii} - 0| \geq \sum_{j \neq i} |M_{ij}|$$

Questo implica che 0 non risiede nel cerchio di centro M_{ii} e raggio $\sum_{j \neq i} |M_{ij}|$ e poiché i è stato scelto in maniera arbitraria non sarà nemmeno nell'unione di questi cerchi, per cui per il teorema dei cerchi di Gershgorin 0 non è un autovalore di $\mathcal{S}^T$. □

Lemma 4.2.2. *Sia $\boldsymbol{p}$ un autovettore della matrice di Google trasposta $\mathcal{G}^T$ di autovalore λ. Sia $|\lambda| > \boldsymbol{\alpha}$. Allora $\boldsymbol{p} = \frac{1}{n}(1 - \boldsymbol{\alpha}) \left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} I_1 \boldsymbol{p}$*

Dimostrazione. Usando la definizione di matrice di Google $\mathcal{G}$ e usando l'ipotesi $\lambda \boldsymbol{p} = \mathcal{G}\boldsymbol{p}$ otteniamo:

$$\lambda \boldsymbol{p} = \mathcal{G}^T \boldsymbol{p} = \left(\boldsymbol{\alpha}\mathcal{S}^T + \frac{1}{n}(1 - \boldsymbol{\alpha}) I_1 \right) \boldsymbol{p} = \boldsymbol{\alpha}\mathcal{S}^T \boldsymbol{p} + \frac{1}{n}(1 - \boldsymbol{\alpha}) I_1 \boldsymbol{p}.$$

Da cui segue

$$\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right) \boldsymbol{p} = \frac{1}{n}(1 - \boldsymbol{\alpha}) I_1 \boldsymbol{p}$$

dal lemma 4.2.1 in quanto abbiamo supposto che $|\lambda| > \boldsymbol{\alpha}$. Si ottiene quindi

$$\boldsymbol{p} = \frac{1}{n}(1-\boldsymbol{\alpha})\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} I_1 \boldsymbol{p}$$

□

Lemma 4.2.3. *Sia λ autovalore della matrice di Google trasposta $\mathcal{G}^T$ e sia $|\lambda| > \boldsymbol{\alpha}$. Allora $\frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}^T\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}\right)^{-1}\boldsymbol{e} = 1$*

Dimostrazione. Sia $\boldsymbol{p}$ un autovettore relativo all'autovalore λ. Cominciamo denotando con $\boldsymbol{e}$ il vettore $n \times 1$ composto da 1 in tutte le entrate. Usando $\boldsymbol{e}$ e ricordando che la matrice I_1 è la matrice con entrate tutte di valore 1 possiamo riscrivere quanto ottenuto nel lemma 4.2.2 in questo modo:

$$\boldsymbol{p} = \frac{(\boldsymbol{e}^T\boldsymbol{p})}{n}(1-\boldsymbol{\alpha})\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1}\boldsymbol{e}$$

Ora supponiamo che $\boldsymbol{e}^T\boldsymbol{p} = 0$, allora per il lemma 4.2.2 $\boldsymbol{p} = 0$, ma questo crea un assurdo in quanto $\boldsymbol{p}$ è un autovettore.

Per cui $\boldsymbol{e}^T\boldsymbol{p} \neq 0$, quindi possiamo definire $\boldsymbol{q} = \frac{\boldsymbol{p}}{\boldsymbol{e}^T\boldsymbol{p}}$. Inoltre $\boldsymbol{q}$ è sempre un autovettore relativo all'autovalore λ e $\boldsymbol{e}^T\boldsymbol{q} = 1$. Date queste ipotesi segue dal lemma 4.2.2 usando la formulazione descritta poco prima che

$$\boldsymbol{q} = \frac{1}{n}(1-\boldsymbol{\alpha})\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1}\boldsymbol{e}$$

E con poche operazioni algebriche segue che:

$$\frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}^T\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}\right)^{-1}\boldsymbol{e} = \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}^T\left(\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}\right)^{-1}\right)^T\boldsymbol{e} =$$

$$\frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}^T\left(\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}\right)^T\right)^{-1}\boldsymbol{e} = \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}^T\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1}\boldsymbol{e} = \boldsymbol{e}^T\boldsymbol{q} = 1$$

□

Questo lemma può essere usato per dare un limite dall'alto a ogni autovalore. Ora abbiamo finalmente tutti gli strumenti per dimostrare il teorema sugli autovalori della matrice di Google:

Dimostrazione. (TEOREMA SUGLI AUTOVALORI DI GOOGLE)

Il primo punto è molto chiaro, $\mathcal{G}$ è una matrice stocastica e, per quanto dimostrato nel lemma 4.1.1, 1 è un autovalore e, in particolare, è il più grande autovalore di $\mathcal{G}$. Dal momento che gli autovalori di una matrice sono gli stessi della sua trasposta il primo punto è dimostrato.

Per il secondo punto useremo l'ultimo lemma dimostrato 4.2.3. Sia λ autovalore della matrice $\mathcal{G}^T$ e sia $|\lambda| > \boldsymbol{\alpha}$. Allora segue che:

$$1 = \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}^T\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}\right)^{-1}\boldsymbol{e} = \frac{(1-\boldsymbol{\alpha})}{n\lambda}\boldsymbol{e}^T\left(I_d - \frac{\boldsymbol{\alpha}}{\lambda}\mathcal{S}\right)^{-1}\boldsymbol{e}.$$

Ora si può dimostrare che $\sum_{n=0}^{\infty} (\frac{\alpha}{\lambda}\mathcal{S})^n$ converge a $(I_d - \frac{\boldsymbol{\alpha}}{\lambda}\mathcal{S})^{-1}$, infatti $\rho\left((\frac{\alpha}{\lambda}\mathcal{S})\right) = \left(\frac{\alpha}{\lambda}\rho\left(\mathcal{S}\right)\right) < \rho\left(\mathcal{S}\right) \leq 1$. L'ultima disuguaglianza segue dal lemma 4.1.1 , in quanto $\mathcal{S}$ è stocastica. Otteniamo quindi:

$$1 = \frac{(1-\boldsymbol{\alpha})}{n\lambda}\boldsymbol{e}^T \sum_{n=0}^{\infty}\left(\frac{\alpha}{\lambda}\mathcal{S}\right)^n \boldsymbol{e} = \frac{(1-\boldsymbol{\alpha})}{n\lambda}\sum_{n=0}^{\infty}\left[\left(\frac{\alpha}{\lambda}\right)^n \boldsymbol{e}^T\mathcal{S}^n\boldsymbol{e}\right]$$

Poiché la matrice $\mathcal{S}$ è stocastica $\mathcal{S}\boldsymbol{e} = \boldsymbol{e}$, applicando questa equazione n volte e ricordando che $\boldsymbol{e}^T\boldsymbol{e} = n$ otteniamo che:

$$1 = \frac{(1-\boldsymbol{\alpha})}{n\lambda}\sum_{n=0}^{\infty}\left[\left(\frac{\alpha}{\lambda}\right)^n \boldsymbol{e}^T\boldsymbol{e}\right] = \frac{(1-\boldsymbol{\alpha})}{n\lambda}\sum_{n=0}^{\infty}\left[\left(\frac{\alpha}{\lambda}\right)^n n\right]$$

Portando fuori la n e usando la serie geometrica otteniamo:

$$1 = \frac{(1-\boldsymbol{\alpha})}{\lambda}\sum_{n=0}^{\infty}\left(\frac{\alpha}{\lambda}\right)^n = \frac{(1-\boldsymbol{\alpha})}{\lambda}\frac{1}{1-\boldsymbol{\alpha}/\lambda} = \frac{1-\alpha}{\lambda-\boldsymbol{\alpha}}$$

Abbiamo quindi ottenuto che $\frac{1-\alpha}{\lambda-\boldsymbol{\alpha}} = 1$, per cui $\lambda = 1$. Ma $\lambda = 1$ è il più grande autovalore della matrice $\mathcal{G}^T$ e $\mathcal{G}$ è primitiva, per cui, per il teorema di Perron, l'autovalore 1 ha una molteplicità algebrica pari a 1. E' quindi impossibile avere $\lambda_1 = \lambda_2 = 1$. Segue che $|\lambda_i| \leq \boldsymbol{\alpha}$ per ogni $2 \leq i \leq n$. □

4.3. Rappresentazione degli autovalori di una matrice di Google

Osserviamo gli autovalori di una matrice:

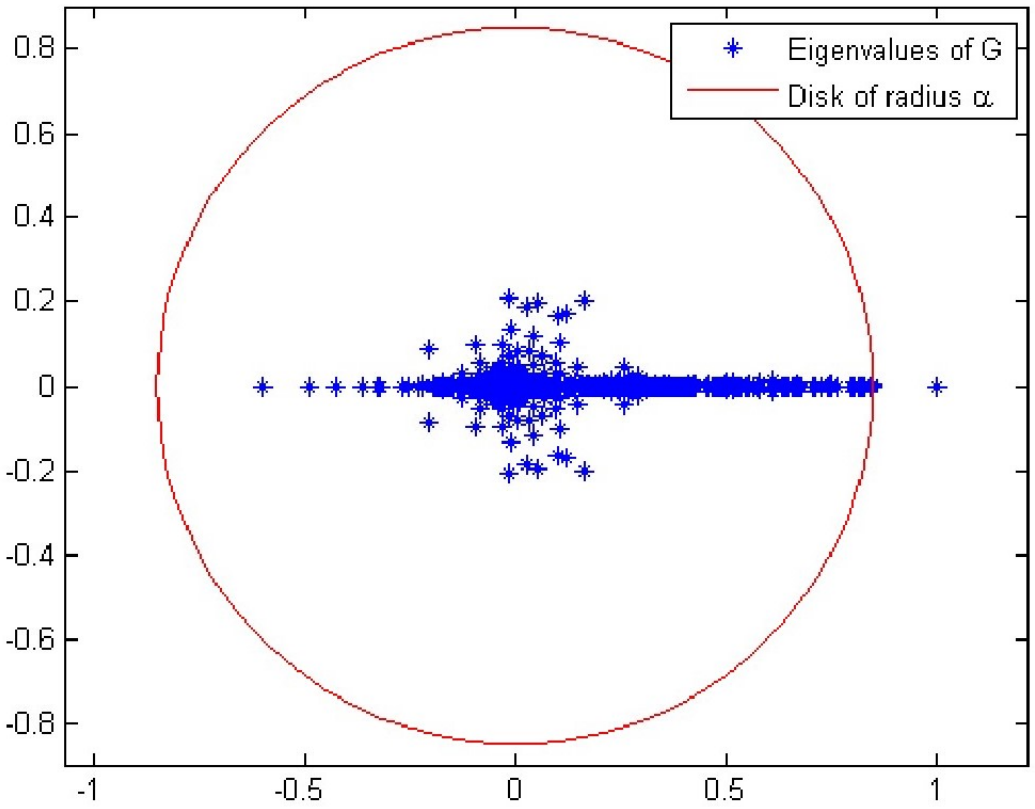

Figura 4.1. Questo grafico rappresenta gli autovalori della matrice di Google di un sottoinsieme del Web composto da tutte le pagine Web i cui link iniziano per www.mathworks.com.

La Figura 4.1 illustra l'ultimo teorema. La figura contiene tutti gli autovalori (in blu) della matrice di Google 5000×5000. Tutti gli autovalori tranne uno sono contenuti nel disco rosso di raggio $\boldsymbol{\alpha}$. L'altro autovalore è esattamente uguale a 1.

Inoltre, se $\mathcal{G}$ ha almeno due classi comunicanti chiuse irriducibili, il secondo autovalore λ_2 è esattamente uguale ad $\boldsymbol{\alpha}$ (per la dimostrazione, si veda [**5**] nella bibliografia). Come vedremo, sapere il valore del secondo autovalore più grande è necessario per calcolare la velocità di convergenza dei vari metodi. Approfondiremo questo punto nel prossimo capitolo.

4.4. Teorema di Perron-Frobenius

In questo paragrafo enunceremo e dimostreremo il teorema di Perron-Frobenius. Tale teorema garantisce l'esistenza e l'unicità del vettore di PageRank, in quanto autovettore di una matrice primitiva rispetto l'autovalore di modulo più grande.

Teorema di Perron-Frobenius.
Sia $\mathcal{T}$ una matrice $n \times n$ irriducibile, non negativa (cioè con tutti gli elementi maggiori o uguali a zero) di periodo h e raggio spettrale $\rho(\mathcal{T}) = r$. Allora:

1. *Il numero $r \in \mathbb{R}^+$ è un numero positivo reale ed è autovalore della matrice $\mathcal{T}$. Viene chiamato l'**autovalore di Perron-Frobenius**.*
2. *L'autovalore di Perron-Frobenius è semplice.*
3. *Esiste un autovettore v della matrice $\mathcal{T}$ rispetto l'autovalore r di norma 1 che ha tutte le componenti positive. Questo autovettore è denominato **autovettore di Perron**.*
4. *Gli unici autovettori con componenti tutte positive sono quelle relative all'autovalore r, ovvero multipli di v.*
5. *Se la matrice $\mathcal{T}$ è primitiva, allora tutti gli autovalori λ di $\mathcal{T}$ soddisfano la seguente proprietà:*
$$r > |\lambda| \tag{4.4.1}$$

Osservazione 4.4.1. *Esiste un risultato più forte di quello nell'ultimo punto. Infatti si può dimostrare che la matrice $\mathcal{T}$ ha esattamente h autovalori di modulo r. Da quest'affermazione possiamo ricondurci all'ultimo punto, infatti, nel caso la matrice $\mathcal{T}$ sia primitiva, sarebbe anche aperiodica, ovvero $h = 1$. Non ci sarebbero, quindi, altri autovalori di modulo pari ad r oltre lo stesso r. Dal momento che r ha molteplicità algebrica 1 e ricordando che r è stato definito come il raggio spettrale della matrice $\mathcal{T}$ abbiamo che $|\lambda_{max}| = \rho(\mathcal{T}) = r > |\lambda|$ con λ qualsiasi altro autovalore della matrice $\mathcal{T}$.*

Osservazione 4.4.2. *Il secondo punto invece mostra che il vettore di PageRank è unico, infatti, l'autovalore di Perron-Frobenius è semplice, ovvero ha molteplicità algebrica 1 e, dal momento che la molteplicità geometrica è minore uguale di quella algebrica, anche quella geometrica è 1. Ricordando che il vettore di PageRank rispetta $\boldsymbol{e}^T\boldsymbol{\pi} = 1$, segue l'unicità. Chiaramente nel caso della matrice di Google 1 è l'autovalore di Perron-Frobenius, in quanto la matrice è stocastica.*

L'esistenza invece segue dal fatto che, data una matrice stocastica, 1 è sempre un suo autovalore, come dimostrato nel lemma 4.1.1.

Per dimostrare il teorema abbiamo bisogno di qualche lemma e teorema preliminare:

Lemma 4.4.1. *Sia A una matrice non negativa irriducibile $n \times n$, allora:*
$$(I_d + A)^{n-1} > 0 \tag{4.4.2}$$

Dimostrazione. Basta dimostrare che $(I_d + A)^{n-1} x > 0$ per ogni $x \geq 0$, x diverso dal vettore nullo.

Sia x un vettore generico tale che $x \geq 0$ e $x \neq 0$. Definiamo la successione di vettori $x^{k+1} = (I_d + A)\, x^k$ per $2 \leq k \leq n-2$ e $x^0 = x$. Poiché A è non negativa e la successione è inizializzata a un vettore x^0 anche esso non negativo, possiamo affermare che x^{k+1} avrà un numero di componenti nulle pari o minore a quelle del vettore x^k.

Ora dimostreremo che il numero di componenti nulle del vettore x^{k+1} è strettamente minore rispetto a quelle del vettore x^k. Supponiamo per assurdo che x^{k+1} e x^k abbiano lo stesso numero di componenti nulle. Quindi esiste una matrice di permutazione P tale che :

$$Px^{k+1} = \begin{bmatrix} y \\ 0 \end{bmatrix} \quad Px^k = \begin{bmatrix} z \\ 0 \end{bmatrix} \text{ con } y, z \in \mathbb{R}^m,\ y, z > 0,\ 1 \leq m < n$$

Allora

$$\begin{pmatrix} y \\ 0 \end{pmatrix} = Px^{k+1} = P\left(x^k + Ax^k\right) = Px^k + PAP^T Px^k =$$

$$= \begin{pmatrix} z \\ 0 \end{pmatrix} + PAP^T \begin{pmatrix} z \\ 0 \end{pmatrix} = \begin{pmatrix} z \\ 0 \end{pmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{pmatrix} z \\ 0 \end{pmatrix}$$

Questo implica che $A_{21} = 0$, ma questo è un assurdo in quanto A è irriducibile. Dal momento che $x = x^0$ ha al massimo $n-1$ elementi nulli, allora x^k avrà al massimo $n-k-1$ elementi nulli, per cui:

$$x^{n-1} = (I_d + A)^{n-1} x^0$$

ha al massimo 0 elementi nulli, per cui è un vettore positivo. Dall'arbitrarietà della scelta di x^0 otteniamo il risultato. □

Sia A una matrice non negativa irriducibile $n \times n$, tale che $A \geq 0$ e sia $x \geq 0$ un vettore non nullo, definiamo la quantità:

$$r_x = \min_{x_i > 0} \left\{ \frac{\sum_{j=1}^n a_{ij} x_j}{x_i} \right\} \tag{4.4.3}$$

Si nota facilmente che r_x è un numero reale non negativo. Inoltre è anche l'estremo superiore dei ρ per cui:

$$Ax \geq \rho x$$

Consideriamo ora la quantità $r = \sup_{x \geq 0, x \neq 0} \{r_x\}$.

Poiché $r_{\alpha x} = r_x$ per ogni $\alpha > 0$ normalizziamo x in modo che $\|x\| = 1$. Allora:

$$r = \sup_{x \in \mathcal{S}} \{r_x\} \quad \text{con } \mathcal{S} = \{x \geq 0 : \|x\| = 1\} \tag{4.4.4}$$

Sia $Q = \left\{ y : y = (I_d + A)^{n-1} x, x \in \mathcal{S} \right\}$. Dal lemma precedente sappiamo che Q è composto solamente di vettori positivi. Otteniamo quindi moltiplicando la disuguaglianza $Ax \geq r_x x$ per $(I_d + A)^{n-1}$ otteniamo:

$$Ay \geq r_x y \quad \text{per cui abbiamo} \quad r_y \geq r_x$$

Quindi la quantità r può essere definita anche in questo modo:

$$r = \sup_{y \in Q} \{r_y\}$$

Poiché $\mathcal{S}$ è un insieme compatto lo è anche Q e poiché r_y esprime una funzione continua sull'insieme Q, esiste necessariamente un vettore z per cui $Az \geq rz$. Chiameremo questi vettori i **vettori estremali** della matrice A.

Lemma 4.4.2. *Sia A una matrice non negativa irriducibile $n \times n$, allora la quantità r definita sopra è positiva. Inoltre tutti i vettori estremali z sono autovettori positivi della matrice A rispetto l'autovalore r.*

Dimostrazione. Per la prima parte consideriamo $r_{\boldsymbol{e}}$, allora:

$$r_{\boldsymbol{e}} = \min_{x_i > 0} \left\{ \frac{\sum_{j=1}^{n} a_{ij} e_j}{e_i} \right\} = \min_{x_i > 0} \left\{ \sum_{j=1}^{n} a_{ij} \right\} > 0 \text{ e } r \geq r_{bme} > 0$$

Per il secondo punto, sia z un vettore estremale tale che $Az - rz = \nu \geq 0$. Supponiamo $\nu \neq 0$, allora, per il lemma 4.4.1 vale:

$$(I_d + A)^{n-1} \nu > 0$$

Da cui, sostituendo la definizione di ν, otteniamo:

$$Aw - rw > 0, \quad w = (I_d + A)^{n-1} z > 0$$

Questo costituisce un assurdo in quanto implica che $r_w > r$. Allora $\nu = 0$, ovvero $Az = rz$. Inoltre $w = (I_d + A)^{n-1} z = (I_d + r)^{n-1} z > 0$, per cui $z > 0$, il che completa la dimostrazione. □

Teorema 4.4.3. *Sia A una matrice non negativa irriducibile $n \times n$ e sia B una matrice complessa $n \times n$ tale che $|B| \leq A$. Allora per ogni autovalore β di B si ha che $|\beta| \leq r$. Inoltre vale l'uguaglianza se e solo se $|B| = A$ e B può essere scritta come $B = e^{i\phi} D A D^{-1}$ con D una matrice diagonale con elementi di modulo 1.*

Dimostrazione. Sia y l'autovettore di B rispetto all'autovalore β, allora per ogni i vale

$$\beta y_i = \sum_{j=1}^{n} b_{ij} y_j.$$

Applicando i moduli vale allora:

$$|\beta||y_i| = \sum_{j=1}^{n} |b_{ij} y_j| \leq \sum_{j=1}^{n} |b_{ij}||y_j| \leq \sum_{j=1}^{n} a_{ij} |y_j|.$$

Ovvero,

$$|\beta||y| \leq |B||y| \leq A|y|$$

che implica che $|\beta| \leq r_{|y|} \leq r$, dimostrando il primo punto.

Se $|\beta| = r$, allora y è un vettore estremale e per il lemma precedente è un autovettore positivo di A rispetto l'autovalore r. Per cui abbiamo che :

$$|\beta||y| = |B||y| = A|y| = r|y|$$

e, poiché $y > 0$, otteniamo che $|B| = A$.

Sia ora D la matrice diagonale di componenti

$$D = \left\{ \frac{y_1}{|y_1|}, ..., \frac{y_n}{|y_n|} \right\}$$

Chiaramente le entrate della matrice D hanno modulo 1 e vale l'equazione $y = D|y|$. Scrivendo β usando l'esponenziale complesso $\beta = re^{i\phi}$ otteniamo:

$$By = \beta y \Rightarrow BD|y| = \beta D|y| \Rightarrow e^{-i\phi}D^{-1}BD|y| = r|y|$$

Sia $C = e^{-i\phi}D^{-1}BD$, allora:

$$C|y| = r|y| = |\beta||y| = B|y| = A|y|$$

e poiché $y > 0$ otteniamo $C = A$, che prova l'ultimo punto. Infine se consideriamo l'altro verso dell'implicazione e prendiamo B in modo che $B = e^{i\phi}DAD^{-1}$, è chiaro che $|B| = A$ e che $|\beta| = r$, concludendo così la dimostrazione. □

Corollario 4.4.4. *Sia A una matrice non negativa irriducibile $n \times n$, allora l'autovalore positivo r è il raggio spettrale di A.*

Dimostrazione. Segue dal teorema 4.4.3 ponendo $B = A$. □

Corollario 4.4.5. *Sia A una matrice non negativa irriducibile $n \times n$ e B una sottomatrice quadrata principale (ovvero che gli elementi sulla diagonale di B sono anche elementi della diagonale di A). Allora $\rho(B) < \rho(A)$.*

Dimostrazione. Poiché B è una sottomatrice quadrata principale di A, esiste una matrice di permutazione P tale che $PA = \begin{bmatrix} B & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.

Consideriamo la matrice $C = \begin{bmatrix} B & 0 \\ 0 & 0 \end{bmatrix}$, chiaramente $C \leq A$, $C \neq A$. Applicando il teorema 4.4.3 alla matrice C al posto di B, otteniamo che $\rho(B) = \rho(C) < \rho(A)$ □

Possiamo finalmente dimostrare il teorema di Perron-Frobenius:

Dimostrazione.

1. $\mathcal{T}$ ha un autovalore positivo di valore uguale al suo raggio spettrale, detto autovalore di Perron-Frobenius:

   vale per il primo corollario 4.4.4.

2. L'autovalore di Perron-Frobenius è semplice:

   Sia $\phi(t)$ il polinomio caratteristico di $\mathcal{T}$, la sua derivata $\phi'(t) = \sum_{i=1}^{n} det\,(\lambda I_d - \mathcal{T}_i)$, dove $\mathcal{T}_i$ è il minore principale di $\mathcal{T}$, ovvero la matrice ottenuta togliendo da $\mathcal{T}$ la riga e la colonna i-esima. Dal corollario 4.4.5 abbiamo che $\rho(A_i) < \rho(A)$ il che implica che $det\,(\lambda I_d - \mathcal{T}_i)$ non si può annullare per qualsiasi $\lambda \geq \rho(A)$. Da questo otteniamo:

   $$det\,(\rho(A) I_d - \mathcal{T}_i) > 0 \Rightarrow \phi'(t) = \sum_{i=1}^{n} det\,(\rho(A) I_d - \mathcal{T}_i) > 0$$

   Il che implica che l'autovalore $\rho(A)$ non può avere molteplicità algebrica superiore a 1.

3. All'autovalore di Perron corrisponde un autovettore y positivo.

   Questo punto vale per il lemma 4.4.2.

4. Gli unici autovettori con componenti tutte positive sono multipli dell'autovettore di y.

   Supponiamo per assurdo che esista un autovettore x non negativo della matrice $\mathcal{T}$ rispetto un autovalore $\lambda \neq r$ che non sia un multiplo di y, ovvero $x \geq 0, x \neq cy$. Moltiplico allora x per $(I_d + A)^k$, con $k \geq n - 1$. Otteniamo quindi che:

   $$(I_d + A)^k y = (I_d + \lambda)^k y > 0 \quad \text{per ogni } k \geq n - 1$$

   Questo implica che

   $$y > 0 \qquad \lambda \in \mathbb{R}^+, \quad \rho(A) > \lambda > -1$$

   Poiché sia x che y sono positivi, esiste un $t > 0$ abbastanza piccolo tale che $z = x - ty \geq 0$. Per cui

   $$Az = A(x - ty) = \rho(A)x - \lambda ty > \rho(A)(x - ty) = \rho(A)z$$

   Questo implica che $r_z > \rho(A)$, che è un assurdo.

5. Se la matrice $\mathcal{T}$ è primitiva, allora tutti gli autovalori λ soddisfano $|\lambda| < r$.

□

Capitolo 5

Calcolare il vettore del PageRank

Il vettore PageRank è l'unico vettore $\boldsymbol{\pi}$ tale che $\boldsymbol{\pi}^{(k+1)} = \mathcal{G}^T\boldsymbol{\pi}^{(k)}$ e tale che $\boldsymbol{\pi}^T\boldsymbol{e} = 1$. Quindi riusciamo a ordinare le pagine Web secondo la loro importanza semplicemente calcolando questo autovettore $\boldsymbol{\pi}$. Il problema è che, come abbiamo già detto, il Web è composto da decine di miliardi di pagine, per cui calcolare l'autovettore non è affatto semplice. Infatti va considerata sia il costo computazionale delle operazioni, sia la memoria necessaria ad effettuare l'algoritmo.

E' per questo motivo che i metodi che andremo a vedere utilizzano le specifiche del problema, in modo da sfruttare al meglio le proprietà della matrice $\mathcal{H}$. Se si trattasse solamente di trovare il vettore di probabilità stazionario rispetto alla catena di Markov di matrice di transizione $\mathcal{G}$ ci sarebbero di sicuro molti altri metodi più complessi e veloci, ma meno adatti al problema.

5.1. Il metodo delle potenze

Il metodo delle potenze è un algoritmo numerico molto conosciuto utile per calcolare l'autovettore relativo all'autovalore più grande. Come metodo è molto lento, infatti la velocità di convergenza dipende dal rapporto fra i due più grandi autovalori della matrice corrispondente, che solitamente è molto piccolo.

Fortunatamente, come abbiamo visto con il Teorema sugli autovalori della matrice di Google 4.2, il secondo autovalore più grande è comunque minore uguale a $\boldsymbol{\alpha}$, il che rende il metodo delle potenze molto efficace per il nostro problema.

Cominciamo mostrando l'algoritmo del metodo delle potenze:

Algoritmo del metodo delle potenze 5.1.1. *Sia $x^{(0)}$ un vettore iniziale. Allora il metodo delle potenze segue questo processo iterativo:*

- $x^{(k+1)} = \mathcal{G}^T x^{(k)}$
- *k=k+1*

Teorema 5.1.1. *Dato un vettore iniziale $x^{(0)}$ che soddisfa $\boldsymbol{e}^T x^{(0)} = 1$ il metodo delle potenze converge sempre al vettore di PageRank $\boldsymbol{\pi}$.*

Dimostrazione. Siano $v_1, ..., v_n$ autovettori corrispondenti alla matrice $\mathcal{G}^T$ che formano una base di $\mathbb{R}^n$ e siano ordinati in modo che i corrispettivi autovalori siano in ordine decrescente.

Riconosciamo allora che, dal momento che la molteplicità algebrica e quindi geometrica dell'autovalore più grande, ovvero dell'autovalore di valore 1, è 1, il vettore v_1 sarà sicuramente uguale a $\boldsymbol{\pi}$ moltiplicato per uno scalare, ovvero $v_1 = \lambda\boldsymbol{\pi}$.

Allora possiamo scrivere $x^{(0)}$ come combinazione lineare di $v_1, ..., v_n$, ovvero $x^{(0)} = \sum_{i=1}^{n} c_i v_i$ con $c_1, ...c_n$ coefficienti in $\mathbb{R}$. Da questo segue che:

$$x^{(0)} = \sum_{i=1}^{n} c_i v_i$$

$$x^{(1)} = \mathcal{G}^T \sum_{i=1}^{n} c_i v_i = \sum_{i=1}^{n} c_i \mathcal{G}^T v_i = \sum_{i=1}^{n} c_i \lambda_i v_i$$

$$\vdots \qquad\qquad \vdots$$

$$x^{(k+1)} = \mathcal{G}^T \sum_{i=1}^{n} c_i \lambda_i^k v_i = \sum_{i=1}^{n} c_i \lambda_i^k \mathcal{G}^T v_i = \sum_{i=1}^{n} c_i \lambda_i^{k+1} v_i$$

Per il teorema sugli autovalori di Google 4.2 sappiamo che, riordinando gli autovalori in ordine decrescente, $\lambda_1 = 1$ e $\lambda_i \leq \boldsymbol{\alpha}$ per ogni $2 \leq i \leq n$. Per cui quando k va a infinito gli autovalori con $2 \leq i \leq n$ tenderanno a 0, in quanto $\boldsymbol{\alpha} < 1$.

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} c_1 \lambda_i^k v_1 = c_1 v_1 = c_1 \lambda \boldsymbol{\pi}$$

Ora mostriamo che $\boldsymbol{e}^T x^{(k)} = 1$ per ogni $k \in \mathbb{N}$. Per ipotesi è vero per $\boldsymbol{e}^T x^{(0)} = 1$, ora supponiamo sia vero per n e mostriamo che sia vero per n+1. Infatti per ipotesi induttiva e sfruttando il fatto che $\mathcal{G}$ è una matrice stocastica abbiamo che:

$$\boldsymbol{e}^T x^{(n+1)} = \boldsymbol{e}^T \mathcal{G}^T x^{(n)} = (\mathcal{G}\boldsymbol{e})^T x^{(n)} = \boldsymbol{e}^T x^{(n)} = 1$$

Dal momento che vale per ogni $i \in \mathbb{N}$ vale anche per il limite, quindi:

$$1 = \lim_{k\to\infty} \boldsymbol{e}^T x^{(k)} = \boldsymbol{e}^T \lim_{k\to\infty} x^{(k)} = c_1 \lambda \boldsymbol{e}^T x^{(k)} = c_1 \lambda$$

Per cui il metodo delle potenze converge effettivamente al vettore di PageRank $\boldsymbol{\pi}$.

Chiaramente non è sempre detto che $x^{(0)}$ possa essere scritto come combinazione lineare di autovettori (ovvero $\mathcal{G}^T$ non è diagonalizzabile). In questo caso scriviamo $\mathcal{G}^T = KJK^{-1}$, dove J è la matrice di Jordan di $\mathcal{G}^T$ e K la matrice corrispondente gli autovettori generalizzati. Ogni blocco J_m di J può essere scritto come $\lambda I_m + N$, dove

N è la matrice composta da tutti zeri eccetto nella diagonale superiore. La matrice N è nilpotente e se $|\lambda| < 1$, $J_m^k \to 0$ per $k \to \infty$. Esiste solamente un autovalore di $\mathcal{G}^T$ che non soddisfa $|\lambda| < 1$, ovvero $\lambda_1 = 1$. Allora il metodo delle potenze convergerà all'autovettore corrispondente a questo autovalore, ovvero il vettore di PageRank.

□

Teorema 5.1.2. *Dato un vettore iniziale $x^{(0)}$ che soddisfa $\boldsymbol{e}^T x^{(0)} = 1$ allora l'errore del metodo delle potenze soddisfa $\|x^{(k)} - \boldsymbol{\pi}\| = \mathcal{O}\left(\boldsymbol{\alpha}^k\right)$.*

Dimostrazione. Siano $v_1, ..., v_n$ come nella dimostrazione precedenti autovettori corrispondenti alla matrice $\mathcal{G}$ che formano una base di $\mathbb{R}^n$ e siano ordinati in modo che i corrispettivi autovalori siano in ordine decrescente e scegliamo come vettore $v_1 = \boldsymbol{\pi}$. Riscriviamo nuovamente $x^{(0)}$ come $x^{(0)} = \sum_{i=1}^n c_i v_i$. Inoltre denotiamo $M = max\{\|c_i v\| : 2 \leq i \leq n\}$.

Dalla dimostrazione precedente sappiamo che $\lim_{k\to\infty} x^{(k)} = \boldsymbol{\pi}$, per cui posso scrivere $\|x^{(k)} - \boldsymbol{\pi}\| \leq \epsilon$. Abbiamo quindi

$$\|x^{(k)} - \boldsymbol{\pi}\| = \|\boldsymbol{\pi} + \sum_{i=2}^{n} c_i \lambda_i^k v_i - \boldsymbol{\pi}\| = \|\sum_{i=2}^{n} c_i \lambda_i^k v_i\| \leq \sum_{i=2}^{n} \|c_i \lambda_i^k v_i\|$$

$$= \sum_{i=2}^{n} |\lambda_i^k| \|c_i v_i\| \leq \sum_{i=2}^{n} \boldsymbol{\alpha} M = (n-1) M \boldsymbol{\alpha}^k = \mathcal{O}\left(\boldsymbol{\alpha}^k\right)$$

□

Il metodo delle potenze ha bisogno di un vettore iniziale, il valore standard per far partire il ciclo è $x^{(0)} = \frac{1}{n}$, chiaramente però se sono note delle preferenze di genere o preferenze per alcune pagine Web, inizializzare il vettore dando più peso a quello pagine porterà probabilmente un risultato migliore. In questo caso il vettore del PageRank viene personalizzato in base all'utente, ma può essere fatta anche una stima più generale.

5.2. Implementazione del metodo delle potenze

Adesso applicheremo il metodo delle potenze, che consiste semplicemente in moltiplicazioni fra matrici e vettori. Il problema è che il prodotto in questione avviene sulla matrice di Google $\mathcal{G}$, che è densa dappertutto. Per questo motivo per il calcolo viene usata la matrice molto sparsa $\mathcal{H}$, infatti ricordando la definizione di matrice di Google e quella di $\mathcal{S}$ otteniamo:

$$\mathcal{G} = \boldsymbol{\alpha}\mathcal{S} + \frac{1}{n}(1-\boldsymbol{\alpha})\mathbb{I}_1 = \boldsymbol{\alpha}\mathcal{H} + \frac{\boldsymbol{\alpha}}{n} a \boldsymbol{e}^T + \frac{1}{n}(1-\boldsymbol{\alpha})\mathbb{I}_1 \tag{5.2.1}$$

Da questo otteniamo

$$x^{(k+1)T} = x^{(k)T}\mathcal{G} = x^{(k)T}\left(\boldsymbol{\alpha}\mathcal{H} + \frac{\boldsymbol{\alpha}}{n} a \boldsymbol{e}^T + \frac{1}{n}(1-\boldsymbol{\alpha})\mathbb{I}_1\right) = \tag{5.2.2}$$

$$= \boldsymbol{\alpha} x^{(k)T} \mathcal{H} + \frac{1}{n} \left(\boldsymbol{\alpha} x^{(k)T} a + 1 - \boldsymbol{\alpha} \right) \boldsymbol{e}^T \tag{5.2.3}$$

Dove nell'ultima equazione abbiamo usato il fatto che $\boldsymbol{e}^T x^{(k)} = 1$.

Poter scrivere in questo modo la matrice di Google fa si che l'unico prodotto vettore matrice sia $\boldsymbol{\alpha} x^{(k)T} \mathcal{H}$, con $\mathcal{H}$ la matrice molto sparsa. In questo modo $\mathcal{S}$ e $\mathcal{G}$ non vengono mai memorizzate, ma basta memorizzare i vettori a e $\boldsymbol{e}$. Ricordiamo che ogni moltiplicazione matrice vettore ha come costo solamente un $\mathcal{O}(n)$, in quanto la matrice $\mathcal{H}$ ha circa solamente 10 elementi diversi da zero per riga.

Ma per quale motivo Brin e Page hanno utilizzato questo metodo e perché è ancora il metodo prevalente usato da Google per il calcolo del vettore di PageRank?

Uno dei motivi più importanti è che non ci sono operazioni sulle matrici, infatti, nonostante $\mathcal{H}$ sia molto sparsa, ci sono comunque circa $10n$ elementi e poiché quella n ha come ordine di grandezza decina di miliardi modificare la matrice richiederebbe un costo computazionale incredibile. Per questo motivo viene preferito ai cosiddetti metodi diretti.

Un'altra caratteristica fondamentale è che questo algoritmo richiede pochissimo spazio di memoria, infatti vanno memorizzate solamente la matrice dei link $\mathcal{H}$, il vettore a e il vettore $x^{(k)T}$ ad ogni iterazione. Per esempio il vettore $x^{(k+1)T}$ è completamente denso, per cui il suo ordine sarà n, ma come già detto n è molto grande e spesso non conviene aumentare la velocità del metodo al costo di aggiungere n spazio di memoria.

L'ultimo motivo importante per cui usare il metodo delle potenze è che si è visto che c'è bisogno solamente dalle 50 alle 100 iterazioni per raggiungere una classifica delle pagine Web soddisfacenti. Dal momento che per ogni iterazione il costo computazionale è di $\mathcal{O}(n)$, è difficile trovare metodi che convergono più velocemente di $50\mathcal{O}(n)$.

Ma perché c'è bisogno di cosi poche iterazioni affinché il metodo converga in maniera soddisfacente? Perché la velocità asintotica di convergenza del metodo è data da $\left\| \frac{\lambda_2}{\lambda_1} \right\|^k$. Sappiamo però che $\lambda_1 = 1$ e $\lambda_2 \leq \boldsymbol{\alpha}$, in più, per come è costruito il grafo del Web è molto probabile che $\lambda_2 \sim \boldsymbol{\alpha}$. Brin e Page hanno usato come valore $\boldsymbol{\alpha} = 0,85$ e sembra che ancora adesso il valore usato sia lo stesso. E' quindi chiaro che dopo 50 iterazioni il tasso di convergenza sarà:

$$\left\| \frac{\lambda_2}{\lambda_1} \right\|^{50} = \boldsymbol{\alpha}^{50} \sim 0.000296.$$

Questo valore garantisce un errore di ordinamento di al massimo 2-3 posti nel vettore di PageRank, che è risultato essere un valore adeguato per il sistema di ricerca di Google.

5.3. Il metodo diretto

A differenza del metodo delle potenze useremo un'altra formulazione simile del problema per calcolare il vettore di PageRank. Infatti invece di calcolare l'autovettore dominante il metodo consiste nel risolvere un sistema lineare. Questa formulazione deriva da un'altra definizione equivalente di vettore di PageRank, ovvero il vettore che soddisfa le seguenti condizioni:

- $\left(I_d - \mathcal{G}^T\right)\boldsymbol{\pi} = 0$
- $\boldsymbol{e}^T\boldsymbol{\pi} = 1$

Dal momento che la matrice $\left(I_d - \mathcal{G}^T\right)$ è a dominanza diagonale alcuni metodi numerici, tra cui quello di Jacobi, convergono. Però usare questa matrice è molto dispendioso di tempo, in quanto la matrice $\mathcal{G}$ ha tutti elementi diversi da zero. Per aiutarci ci incorre il seguente teorema, diretta conseguenza del lemma 4.2.2:

Teorema 5.3.1. *Il vettore di PageRank $\boldsymbol{\pi}$ soddisfa la seguente equazione:*

$$\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)\boldsymbol{\pi} = \frac{1}{n}\left(1 - \boldsymbol{\alpha}\right)\boldsymbol{e}. \tag{5.3.1}$$

Dimostrazione. Basta usare il risultato del lemma 4.2.2, ovvero:

$$\boldsymbol{p} = \frac{1}{n}(1 - \boldsymbol{\alpha})\left(\lambda I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} I_1\boldsymbol{p}$$

e porre $\boldsymbol{p} = \boldsymbol{\pi}$ e $\lambda = 1$, ricordando che $\boldsymbol{e}^T\boldsymbol{\pi} = 1$. □

La matrice $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)$ ha delle importanti proprietà:

- tutti gli autovalori di $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)$ sono nel disco di raggio $\boldsymbol{\alpha}$ e centrati in 1. Ciò segue direttamente dal teorema di Gershgorin.
- $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)$ è invertibile, in quanto è stato dimostrato nel lemma 4.2.1.
- $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)$ è una matrice a dominanza diagonale stretta.
- la somma delle righe di $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)$ è esattamente $\boldsymbol{\alpha}$.

Possiamo semplificare ulteriormente il teorema appena visto 5.3:

Teorema 5.3.2. *Il vettore di PageRank $\boldsymbol{\pi}$ soddisfa la seguente equazione:*

$$\left(I_d - \boldsymbol{\alpha}\mathcal{H}^T\right)\boldsymbol{\pi} = \frac{1}{n}\boldsymbol{e} \tag{5.3.2}$$

Dimostrazione. Prendiamo il risultato dell'ultimo teorema e sostituiamo ad $\mathcal{S}$ la sua definizione:

$$\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)\boldsymbol{\pi} = \frac{1}{n}\left(1-\boldsymbol{\alpha}\right)\boldsymbol{e}$$

$$\left(I_d - \boldsymbol{\alpha}\mathcal{H}^T - \frac{\boldsymbol{\alpha}}{n}a\boldsymbol{e}^T\right)\boldsymbol{\pi} = \frac{1}{n}\left(1-\boldsymbol{\alpha}\right)\boldsymbol{e}$$

Portiamo al membro destro $-\frac{\boldsymbol{\alpha}}{n}a\boldsymbol{e}^T\boldsymbol{\pi}$ e ricordando che $\boldsymbol{e}^T\boldsymbol{\pi} = 1$ otteniamo.

$$\left(I_d - \boldsymbol{\alpha}\mathcal{H}^T\right)\boldsymbol{\pi} = \frac{1}{n}\left(1-\boldsymbol{\alpha}\right)\boldsymbol{e} + \frac{\boldsymbol{\alpha}}{n}a$$

□

Osservazione 5.3.1. *Le proprietà della nuova matrice* $\left(I_d - \boldsymbol{\alpha}\mathcal{H}^T\right)$ *sono le stessa della matrice* $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)$*, ad eccezione della somma delle righe, che può essere 1 o* $1-\boldsymbol{\alpha}$*.*

5.4. Metodo di Jacobi

Gli ultimi due teoremi ci permettono di calcolare il vettore di PageRank in un nuovo modo. Useremo entrambi i teoremi appena discussi e il metodo di Jacobi per approssimare il vettore di PageRank. Questo metodo sfrutta il fatto che

$$\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} = \sum_{n=0}^{\infty}\left(\boldsymbol{\alpha}\mathcal{S}^T\right)^n$$

Questa somma converge perché $\rho(\boldsymbol{\alpha}\mathcal{S}^T) = \boldsymbol{\alpha}\rho(\mathcal{S}^T) \leq \boldsymbol{\alpha}\cdot 1 < 1$. Approssimeremo questa serie con una somma parziale, ma anche con un numero basso di iterazioni i prodotti matrice per matrice sono molto dispendiosi computazionalmente. Per questo motivo useremo il metodo di Jacobi.

Algoritmo del metodo delle potenze 5.4.1. *L'algoritmo di Jacobi applicato alla matrice* $\mathcal{S}$*:*

```
k=0;
x0=(1-alpha)/n;
x0=x0*ones(n,1);
xnew=x0;
while (non viene raggiunta la convergenza necessaria)
    xold=xnew
    xnew=alpha*S'*xold+x0:
    k=k+1;
end
```

Ovvero itera:

$$\begin{cases} x^{(0)} = \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e} \\ x^{(k+1)} = \boldsymbol{\alpha}\mathcal{S}^T x^{(k)} + \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e} \end{cases} \tag{5.4.1}$$

fino a raggiungere la convergenza.

Teorema 5.4.1. *Il metodo di Jacobi converge al vettore di PageRank* $\boldsymbol{\pi}$*. Inoltre, l'errore dopo* k *iterazioni è dell'ordine di* $\mathcal{O}(\boldsymbol{\alpha}^k)$*.*

Dimostrazione. E' facile notare per induzione che $x^k = \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{k} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}$.

Sia $k = 0$: $x^0 = \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e} = \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{0} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}$.

Supponiamo sia vero per $k \in \mathbb{N}$, allora

$$x^{k+1} = \boldsymbol{\alpha}\mathcal{S}^T x^k + \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e} = \boldsymbol{\alpha}\mathcal{S}^T \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{k} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e} + \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e} =$$

$$= \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=1}^{k+1} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e} + \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e} = \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{k+1} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}$$

Per cui vale l'uguaglianza per ogni $k \in \mathbb{N}$. Per k che tende all'infinito,

$$\frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{k} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e} \to \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{\infty} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e} = \frac{1}{n}(1-\boldsymbol{\alpha}) \left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} \boldsymbol{e}$$

Allora, per il teorema 5.3.1, x^k per k che tende a infinito tende a $\boldsymbol{\pi}$.

Dopo k iterazioni, l'errore di $\mathbf{x}$ è uguale a:

$$\left\|\boldsymbol{\pi} - x^k\right\| = \left\|\boldsymbol{\pi} = \frac{1}{n}(1-\boldsymbol{\alpha}) \left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} \boldsymbol{e} - \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=0}^{k} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}\right\|$$

$$= \frac{(1-\boldsymbol{\alpha})}{n} \left\|\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)^{-1} \boldsymbol{e} - \sum_{n=0}^{k} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}\right\|$$

$$= \frac{(1-\boldsymbol{\alpha})}{n} \left\|\sum_{n=0}^{\infty} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e} - \sum_{n=0}^{k} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}\right\|$$

$$= \frac{(1-\boldsymbol{\alpha})}{n} \left\|\sum_{n=k+1}^{\infty} (\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}\right\| \leq \frac{(1-\boldsymbol{\alpha})}{n} \sum_{n=k+1}^{\infty} \left\|(\boldsymbol{\alpha}\mathcal{S}^T)^n \boldsymbol{e}\right\|$$

$$\leq \frac{(1-\boldsymbol{\alpha})\boldsymbol{\alpha}^k}{n(1-\boldsymbol{\alpha})} \|\boldsymbol{e}\| = \boldsymbol{\alpha}^k = \mathcal{O}(\boldsymbol{\alpha}^k),$$

il che completa la dimostrazione. □

Osserviamo che possiamo applicare il metodo di Jacobi anche alla matrice $\boldsymbol{\alpha}\mathcal{H}^T$ per approssimare $\boldsymbol{\pi}$, la convergenza è assicurata dal teorema 5.3.2. Possiamo quindi usare anche questo algoritmo:

Algoritmo del metodo delle potenze 5.4.2. *L'algoritmo di Jacobi applicato alla matrice $\mathcal{H}$:*

```
k=0;
x0=ones(n,1)/n;
xnew=x0;
while (non viene raggiunta la convergenza necessaria)
    xold=xnew
    xnew=alpha*H'*xold+x0;
    k=k+1;
end
```

Ovvero itera:

$$\begin{cases} x^{(0)} = \frac{1}{n}\boldsymbol{e} \\ x^{(k+1)} = \boldsymbol{\alpha}\mathcal{H}^T x^{(k)} + \frac{1}{n}\boldsymbol{e} \end{cases} \tag{5.4.2}$$

fino a quando il metodo converge.

Usando il teorema 5.3.2 possiamo dimostrare, nello stesso modo in cui abbiamo dimostrato il teorema 5.4.1 che anche questo metodo converge a $\boldsymbol{\pi}$ e che l'errore, dopo k iterazioni, sarà pari a $\mathcal{O}(\boldsymbol{\alpha}^k)$. Il vantaggio di questo metodo è che la matrice $\mathcal{H}$ è molto più sparsa di $\mathcal{S}$, per cui i prodotti fra matrice e vettore saranno più veloci.

Capitolo 6

Analisi dei dati

Nel capitolo precedente abbiamo discusso sui vari metodi possibili. In questo capitolo mostreremo i risultati dell'applicazione di questi metodi e li confronteremo usando vari data base.

6.1. Dati

Prima di tutto è necessario avere dei dati per costruire la matrice $\mathcal{H}$. Chiaramente non conosciamo l'intero grafo che descrive tutte le pagine Web, per questo motivo facciamo uso di database più piccoli che differiscono di alcune proprietà. In questo modo possiamo vedere anche quali metodi preferiscano dei database con proprietà specifiche.

Fortunatamente molte persone hanno già costruito dei grafi su porzioni del Web e le hanno rese pubbliche. Una di queste matrici forma una raccolta di siti Web che riguardano la California. Si tratta di circa 10 mila pagine e circa il 50 per cento di queste sono nodi senza outer link. Ringraziamo Jon Kleinberg per questo set di dati, si veda [**6**] nella bibliografia.

Abbiamo utilizzato anche una matrice di link di dimensione molto più grande. Si tratta di una raccolta di tutte le pagine Web (quasi 300 mila) del sito dell'Università di Stanford. Ringraziamo Sep Kamvar per questi set di dati, si veda [**7**] nella bibliografia. Vale la pena di notare che questa matrice più grande ha un numero relativamente piccolo di nodi senza outer links (su 300 mila pagine Web solamente 10 mila non hanno outer link). Ciò avrà conseguenze sul tempo di iterazione di ogni metodo numerico.

6.2. Criteri di Convergenza

I metodi numerici che abbiamo discusso sono iterativi. Le iterazioni devono essere interrotte quando viene raggiunta una approssimazione soddisfacente del vettore di PageRank. Ma quand'è questo il caso? Sia $\|\cdot\|$ una norma. Diremo che l'approssimazione $\mathbf{x}$ di $\boldsymbol{\pi}$ è soddisfacente quando il residuo relativo è abbastanza piccolo,

ovvero:

$$\frac{\|\mathcal{G}^T\mathbf{x} - \mathbf{x}\|}{\|\mathbf{x}\|} < \epsilon \tag{6.2.1}$$

per un qualche ϵ abbastanza piccolo $\epsilon > 0$. Per fare questi esperimenti abbiamo preso $\epsilon = 10^{-5}$ e come norma $\|\cdot\|$ la norma 1. Scegliendo $\epsilon = 10^{-5}$ ci assicuriamo che il vettore di PageRank sia sbagliato di al massimo una posizione e anche questo accade molto raramente. Infatti, usando il data base sull'università di Stanford e usando quel valore di epsilon, abbiamo misurato in totale meno di 100 errori di posizione (su 300 mila) del vettore di PageRank dopo averlo confrontato con il vettore calcolato con precisione di $\epsilon = 10^{-10}$. Questo perché quello che ci interessa è l'ordine dei valori del vettore di PageRank, non il valore esatto che viene restituito ad ogni pagina, per cui porre un ϵ troppo piccolo non porta alcun beneficio. La scelta della norma 1 è stata fatta per lo stesso motivo.

- Sia $\mathbf{x}^{(i)}$ l'approssimazione di $\boldsymbol{\pi}$ usando il metodo delle potenze all'i-esima iterazione. Il metodo delle potenze sarà ripetuto finché $\frac{\|\mathcal{G}^T\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\|}{\|\mathbf{x}\|} < \epsilon$, ovvero fino a

$$\boldsymbol{e}^T|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}| < \epsilon \quad \text{ricordando che } \boldsymbol{e}^T\mathbf{x}^{(j)} = 1 \text{ per ogni } j.$$

- Per il metodo di Jacobi (applicato alla matrice $\mathcal{S}$) dopo i iterazioni abbiamo un'approssimazione $x^{(i)}$ per cui $\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)x^{(i)} \sim \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}$. Le iterazioni saranno ripetute finché:

$$\left\|\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)x^{(i)} - \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}\right\| < \epsilon$$

  ovvero

$$\boldsymbol{e}^T\left|\left(I_d - \boldsymbol{\alpha}\mathcal{S}^T\right)x^{(i)} - \frac{(1-\boldsymbol{\alpha})}{n}\boldsymbol{e}\right| < 10^{-5}$$

- Per il metodo di Jacobi (applicato alla matrice $\mathcal{H}$, dobbiamo normalizzare $x^{(i)}$ per avere un'approssimazione di $\boldsymbol{\pi}$. Per cui le iterazioni sono ripetute finché $\left\|\boldsymbol{\alpha}\mathcal{H}^T x^{(i)} - \frac{1}{n}\boldsymbol{e}\right\| / (\boldsymbol{e}^T x^{(i)}) < \epsilon$, ovvero

$$\boldsymbol{e}^T\left|\boldsymbol{\alpha}\mathcal{H}^T x^{(i)} - \frac{1}{n}\boldsymbol{e}\right| < 10^{-5}\boldsymbol{e}^T x^{(i)}$$

Comunque per ogni algoritmo è stato fissato un numero massimo di iterazioni uguale a 1000. in modo da evitare che il metodo continui all'infinito quando viene messa in input una matrice dei link che non soddisfa i criteri iniziali.

6.3. Analisi dati

In questo paragrafo mostreremo i risultati numerici dei 3 metodi visti nel capitolo precedente, confrontando i loro tempi di computazione su vari data set e mostrando il numero di iterazione di ogni metodo. Inoltre mostreremo cosa accade quando viene cambiato il valore di $\boldsymbol{\alpha}$.

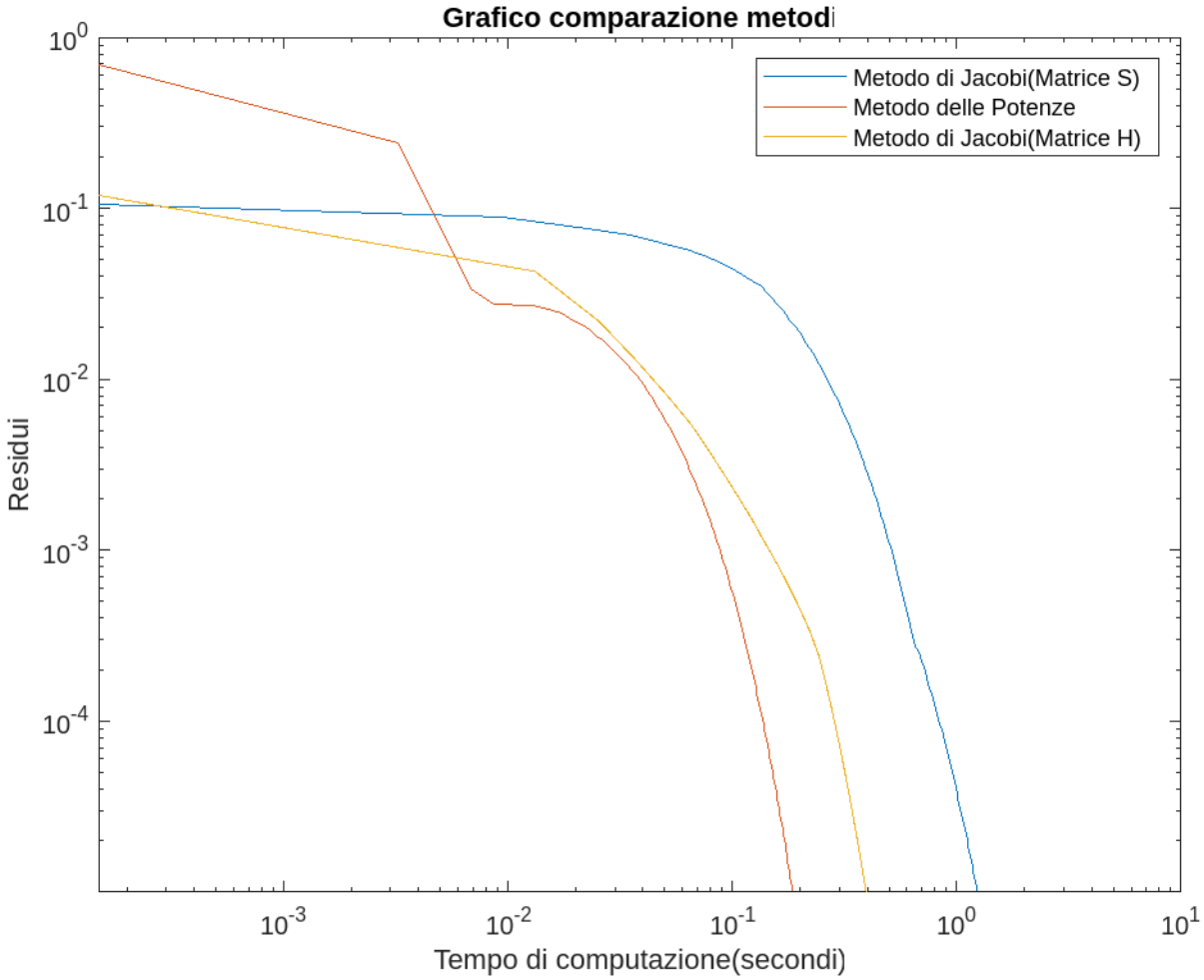

Figura 6.1. I 3 metodi numerici comparati usando come data set la porzione del Web sulla California, si veda [**6**] nella bibliografia. ($\boldsymbol{\alpha} = 0.85$)

La figura 6.1 rappresenta sull'asse delle y il residuo, calcolato per ogni metodo come descritto nel paragrafo precedente. Sull'asse delle x, invece, è rappresentato il tempo trascorso ad ogni iterazione in secondi. Il grafico è in scala logaritmica sia per le ascisse che per le ordinate.

Facciamo ora delle osservazioni usando lo stesso data set ma cambiando il valore del parametro di teletrasporto $\boldsymbol{\alpha}$.

Tabella 6.1. I 3 metodi confrontati sullo stesso data set sulla California ma con vari valori di $\boldsymbol{\alpha}$. Viene rappresentato nella tabella il numero di iterazioni e il tempo di computazione per ogni metodo in secondi.

$\boldsymbol{\alpha}$	Metodo delle Potenze	Metodo Jacobi ($\mathcal{H}$)	Metodo Jacobi ($\mathcal{S}$)
$\boldsymbol{\alpha}$=0.50	0.3234 12 Iterazioni	0.4371 11 Iterazioni	0.2536 15 Iterazioni
$\boldsymbol{\alpha}$=0.70	0.5157 23 Iterazioni	0.6485 20 Iterazioni	0.6371 28 Iterazioni
$\boldsymbol{\alpha}$=0.85	0.9949 47 Iterazioni	1.1052 42 Iterazioni	1.4058 59 Iterazioni
$\boldsymbol{\alpha}$=0.95	3.5275 142 Iterazioni	3.1919 128 Iterazioni	3.9975 166 Iterazioni

Come si può notare, all'avvicinarsi del parametro $\boldsymbol{\alpha}$ a 1, il metodo di Jacobi rispetto la matrice $\mathcal{H}$ prende il sopravvento, superando la velocità di convergenza del metodo delle Potenze. Inoltre osserviamo come il metodo di Jacobi applicato a $\mathcal{S}$ sia sempre il metodo più lento. Infine da questi esperimenti numerici possiamo osservare come cambia il numero di iterazioni al variare del parametro di teletrasporto $\boldsymbol{\alpha}$. Segue, infatti, quanto avevamo dimostrato nel capitolo precedente, ovvero che l'errore ad ogni k-esima iterazione è $\mathcal{O}(\boldsymbol{\alpha}^k)$, per cui ci vorranno circa $k = \log_{\boldsymbol{\alpha}}(\epsilon)$ iterazioni per la convergenza del metodo.

Il data base sulla California è composto di moltissime pagine Web senza link in uscita, infatti sono ben 4637, ovvero quasi la metà delle pagine Web (9663). Consideriamo ora il caso in cui il data base quasi non contenga pagine senza link in uscita, considerando la porzione di Web composta da tutte le pagine relative all'università di Stanford.

Il data base di Stanford è di gran lunga più grande rispetto quello sulla California, conta, infatti, quasi 300 mila pagine Web. Questo implica che per implementare il metodo su Matlab è necessario usare comandi che non immagazzinano gli elementi nulli della matrice, altrimenti Matlab non ti lascerà compilare. Per fare questo tutte le variabili che utilizziamo saranno variabili double sparse. Inoltre il data set di Stanford restituisce una matrice che esprime anche il numero di link che mandano dalla pagina i alla j. E' stato quindi necessario normalizzare questa matrice e, viste le dimensioni, anche questo passaggio è sostanziale quando si considera il tempo di computazione. Infatti, per costruire questa matrice, ci sono voluti circa 10 minuti. Mostriamo ora cosa accade per $\boldsymbol{\alpha} = 0.85$:

Metodo delle Potenze	Metodo Jacobi ($\mathcal{H}$)	Metodo Jacobi ($\mathcal{S}$)
1.7087 58 Iterazioni	1.568 50 Iterazioni	1.882 60 Iterazioni

Osserviamo che nel caso in cui ci siano pochi nodi senza outer link il metodo di Jacobi su $\mathcal{H}$ prende il sopravvento sul metodo delle potenza.

Per quanto osservato con il data set precedente ci aspettiamo che all'aumentare del parametro di teletrasporto questo distacco sulla velocità di convergenza aumenti:

α	Metodo delle Potenze	Metodo Jacobi ($\mathcal{H}$)	Metodo Jacobi ($\mathcal{S}$)
α=0.90	2.9948 87 Iterazioni	2.5861 76 Iterazioni	3.0140 88 Iterazioni
α=0.95	5.5547 165 Iterazioni	4.8141 150 Iterazioni	5.7978 166 Iterazioni

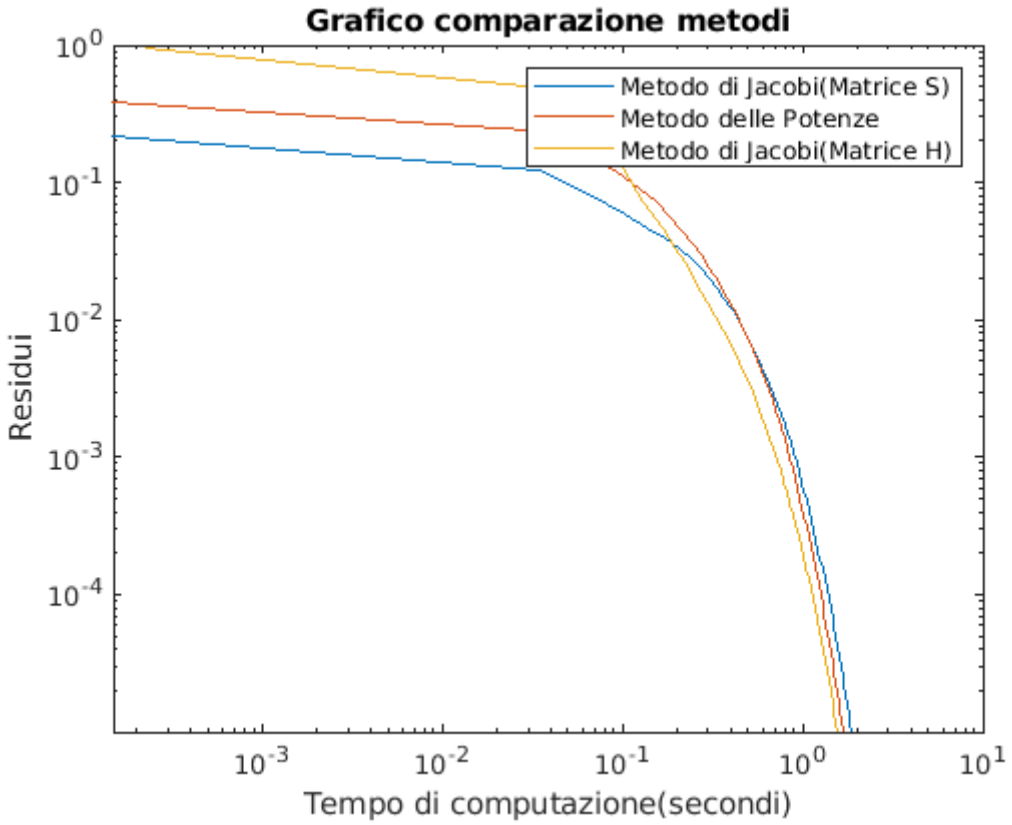

Figura 6.2. Comparazione dei 3 metodi usando come data set la porzione del Web riguardo l'università di Stanford e $\alpha = 0.85$.

6.4. Discussione

Come abbiamo visto nel capitolo precedente, ogni metodo numerico ha un tasso di convergenza di $\mathcal{O}(\alpha^k)$. Pertanto, non è troppo inaspettato vedere che la velocità dei tre metodi è circa la stessa. Nella maggior parte dei casi, il metodo della potenza è l'algoritmo più veloce. Tuttavia, la differenza tra questo metodo e il Metodo di Jacobi applicato ad $\mathcal{H}$ è molto piccola. In alcuni casi, ad esempio se la quantità di nodi senza link in uscita in una rete è elevata e quando consideriamo α vicino a 1, quest'ultimo metodo convergerà più velocemente.

Capitolo 7

Bibliografia

[**1**] L. Page, S. Brin, R. Motwani and T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web. 1998

[**2**] A. Langville and C. Meyer, Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, New Jersey, 2006

[**3**] C. Moler, The worlds largest matrix computation. Matlab news and notes, 2002

[**4**] A. Langville and C. Meyer, Deeper Inside PageRank. Internet Mathematics, 2004

[**5**] T. Haveliwala and S. Kamvar, The Second Eigenvalue of the Google Matrix. Stanford University Technical Report, 2003

[**6**] J. Kleinberg, California hyperlink matrix. http://www.cs.cornell.edu/Courses/cs685/2002fa/

[**7**] https://www.cise.ufl.edu/research/sparse/matrices/listbydimension.html

[**8**] Dimitrios Noutsos, Perron Frobenius theory and some extensions

Capitolo 8

Appendice

8.1. Costruzione della matrice dei link $\mathcal{H}$ sul data base California

```
format long;
%Legge il file http://www.cs.cornell.edu/courses/cs685/2002fa/data/gr0.California
%Con i dati costruisce la matrice dei link
% Apri il file in lettura
fileID = fopen('caif.txt', 'r');

% Inizializza vettori per le colonne
colonna1 = [ ];
colonna2 = [ ];

while ~feof(fileID)
    line = fgetl(fileID); % Leggi una riga come stringa
        if ischar(line)
            % Converti la riga in un array di numeri
            data = str2double(split(line));
        if numel(data) >= 2 % Assicurati che ci siano almeno due numeri
            % Salva il primo numero nella colonna1
            colonna1(end+1) = data(2)+1;
            % Salva il secondo numero nella colonna2
            colonna2(end+1) = data(3)+1;
        end
    end
end
colonna1(end+1)=0;
colonna2(end+1)=0;
% Chiudi il file
fclose(fileID);
```

```
n=9663
k=0; j=1;
%Costruzione matrice dei link (9663 pagine Web)
H=zeros(n);
for i=1:n
      k=0;
     while colonna1(j)==i
           j=j+1;
           k=k+1;
      end
      H(i,colonna2(j-k:j-1))=1/k;
end
```

8.2. Costruzione della matrice dei link $\mathcal{H}$ sul data base Stanford

```
S = load("Stanford(1).mat"); A = S.Problem.A;

A=A-diag(diag(A));

% Prende la matrice e rende ogni elemento diverso da 0 pari a 1
A=logical(A);

% Conta il numero di elementi non nulli
g=nnz(A);

% Trova gli indici in cui la matrice è diversa da zero

[indici_righe, indici_colonne] = find(A);

% Combina gli indici in una matrice

indici = [indici_colonne, indici_righe];

% Ordina gli indici per riga in ordine crescente

indici_ordinati = sortrows(indici, 1);

% Estrai gli indici ordinati delle righe e delle colonne

indici_righe_ordinate = indici_ordinati(:, 1);
indici_colonne_ordinate = indici_ordinati(:, 2);

% Rendi i vettori riga vettori colonna

indici_righe_ordinate = vertcat(indici_righe_ordinate, 0);
indici_colonne_ordinate = vertcat(indici_colonne_ordinate, 0);

clear A;

n=281903;

H=sparse(n);
```

```
k=0;j=1;

for i=1:n
    k=0;
    while indici_righe_ordinate(j)==i
        j=j+1
        k=k+1;
    end
    H(i,indici_colonne_ordinate(j-k:j-1))=1/k;
end

H=H-diag(diag(H));

clear indici_righe_ordinate;

clear indici_colonne_ordinate;
```

8.3. Codice che implementa i 3 metodi

```
%Costruisco il vettore a, ovvero il vettore che dà 1 se la pagina non ha link uscenti
a=(sum(H,2)==0);


%Funzione che restituisce la norma 1
norm=@(x)sum(abs(x));


%Errore massimo
error=1e-5;


% Vettore iniziale
v=e/n;


%Parametro di teletrasporto
alpha=0.85;


%Numero massimo di iterazioni
max=1000;


K=alpha*H';


e=ones(n,1);


%Metodo di Jacobi con matrice H
TempoPerIterH=[]; ErrPerIterH=[];
pih=v;
```

```
alphahpi=K*pih;
tic
for iterh=2:max
    % Salvo tempo per iterazione
    TempoPerIterH(end+1)=toc;
    pih=alphahpi+v;
    alphahpi=K*pih;
    %Calcolo il residuo e lo salvo
    resh=norm(pih-alphahpi-v);
    resh=resh/sum(pih);
    ErrPerIterH(end+1)=resh;
    %Condizione per interrompere l'algoritmo
    if(resh<error)
        break;
    end
end
pih=pih/sum(pih);


%Applica il metodo delle Potenze
TempoPerIterP=[]; ErrPerIterP=[];
pip=v;
tic
for iterp=2:max
    %Salvo tempo per iterazione
    TempoPerIterP(end+1)=toc;
    piprevious=pip;
    pip=K*pip+(1-alpha+alpha*sum(d.*pip))*v;
    %Calcolo il residuo e lo salvo
    resp=norm(pip-piprevious);
    ErrPerIterP(end+1)=resp;
    %Condizione per interrompere l'algoritmo
        if(resp<error)
    break;
    end
end


%Applica il metodo di Jacobi con matrice S
TempoPerIterS=[]; ErrPerIterS=[];
pis=(1-alpha)*v;
alphaspi=K*pis+pis;
tic
for iters=2:max
    %Salvo tempo per iterazione
    TempoPerIterS(end+1)=toc;
    pis=alphaspi+(1-alpha)*v;
    alphaspi=K*pis+sum(d.*pis)*alpha*v;
```

```
    %Calcolo il residuo e lo salvo
    resS=norm(pis-alphaspi-(1-alpha)*v);
    ErrPerIterS(end+1)=resS;
    %Condizione per interrompere l'algoritmo
        if(resS<error)
    break;
    end
end
```

8.4. Codice che plotta i grafici dei 3 metodi

```
%Costruisce il grafico
figure;
loglog(TempoPerIterS,ErrPerIterS,TempoPerIterP,ErrPerIterP,TempoPerIterH,ErrPerIterH);
legend('Metodo di Jacobi(Matrice S)','Metodo delle Potenze','Metodo di Jacobi(Matrice H)');
xlim([0.00015,10]);
ylim([1e-5,1]);1
xlabel('Tempo di computazione(secondi)');
ylabel('Residui');
title('Grafico comparazione metodi');
```