Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

STRATEGIE DI ACCELERAZIONE PER L'APPROSSIMAZIONE DEL VETTORE DI PAGERANK

Relatore:
Chiar.ma Prof.
Valeria Simoncini

Presentata da:
Laura Cavalli

II Sessione
Anno Accademico 2020/21

Alla mia famiglia, a Davide,
alle mie coinquiline, compagne di questa nuova vita
e alle mie amiche, compagne della vecchia,
a tutte le bocciature che mi hanno fatto crescere.
A tutto ciò che mi ha portato fin qui.

Indice

Introduzione

Il World Wide Web conta ai giorni d'oggi circa 50 miliardi di pagine web [1]. Nonostante questa mole di dati, non c'è domanda che rimanga senza una risposta estremamente accurata per più di qualche frazione di secondo.
Com'è stato dunque possibile dare ordine a questo caos?
La chiave di tutto sono i motori di ricerca che non solo filtrano il World Wide Web per una specifica richiesta, ma soprattutto ordinano le pagine Web per importanza.
Google fu il primo motore di ricerca che riuscì a classificare le pagine web per importanza grazie al metodo di PageRank sviluppato da Larry Page e Sergey Brin nel 1996. Loro stessi nel famoso trattato "Bringing order to the web " definiscono il PageRank come "un metodo per classificare le pagine web in maniera oggettiva e meccanica, misurando efficacemente l'interesse umano e l'attenzione loro dedicata." [1] .
L'obiettivo di questa tesi di laurea è quello di approfondire l'aspetto matematico che si nasconde dietro tutto ciò, soffermandosi sui due diversi aspetti che questo problema può assumere: mentre da una parte si riduce al calcolo dell'autovalore più grande della matrice di Google, dall'altra si può presentare come risoluzione di un sistema lineare.
La tesi è dunque articolata in 4 capitoli:
Nel primo capitolo verranno date le prime definizioni e introdotti tutti gli strumenti usati in seguito. Nel secondo capitolo si analizzeranno i metodi basati sul problema agli autovalori, partendo dal classico metodo delle potenze per poi esplorarne le recenti implementazioni. Nel terzo capitolo verrà trattato il problema dal punto di vista della risoluzione di un sistema lineare, partendo da Jacobi e approfondendo un nuovo metodo basato sull'algoritmo di Arnoldi. Infine, dopo aver analizzato i dati raccolti dagli esperimenti numerici, si metteranno a confronto i vari metodi studiati per trarne delle interessanti conclusioni.

[1] fonte: https://www.worldwidewebsize.com/

Capitolo 1

La matematica dietro il PageRank

Come già accennato nell'introduzione, il metodo di Page Rank è un algoritmo basato sull'assegnazione di un livello di importanza ad ogni pagina web.
Nei capitoli seguenti vedremo come questo punteggio venga determinato.

1.1 Struttura a link del Web

Definizione 1 (Grafo diretto)**.**
Un grafo diretto (o grafo orientato) G è una coppia (V,E) dove

- V è un insieme di nodi (o vertici);
- E $\subseteq$ V x V è un insieme di archi.

Possiamo considerare il World Wide Web come un grande grafo diretto dove i nodi rappresentano le pagine web e gli archi da un nodo i ad un nodo j i collegamenti che dalla pagina P_i portano alla pagina P_j.
Ora, se molte pagine puntano verso una stessa pagina è intuitivo vedere che la pagina "puntata" sia importante. Tuttavia la *quantità* di link verso una determinata pagina è facilmente manipolabile quindi Brin e Page si concentrarono sulla *qualità* dei link in uscita; pertanto, se una pagina ritenuta importante indirizza verso un'altra pagina, allora anche quest'ultima sarà ritenuta importante.

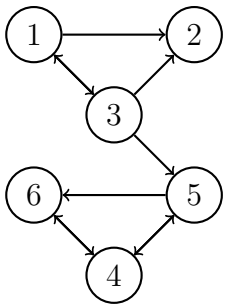


Figura 1.1: Grafo diretto rappresentante un web di 6 pagine [2]

Un modo alternativo per visualizzare il problema è dato dal concetto di **random surfer** introdotto da Brin e Page: possiamo immaginare un utente che navighi in modo randomico per il web; quando giunge ad un nodo con più outlinks esso sceglie che link seguire in maniera casuale e continuerà in questo modo indefinitamente. A lungo andare, la quantità di tempo in cui il random surfer si sofferma su una determinata pagina può essere vista come una misura dell'importanza di quella pagina; infatti se vi passa molto tempo è probabile che nel suo cammino aleatorio abbia percorso svariati link che lo abbiano riportato proprio su quella pagina.
Pertanto, anche in questo metodo, quella pagina può essere considerata importante poiché puntata da molte altre pagine importanti.

1.2 Page Rank come somma pesata

Fino ad ora abbiamo ben appreso il motto del Page Rank : "una pagina è importante se è puntata da altre pagine importanti". Proviamo dunque a spiegare questo concetto in un modo più matematico.

Definizione 2.
Fissata una pagina P_i chiamiamo

- outlink un link che partendo da P_i punta verso altre pagine
- inlink un link che punta verso P_i .

Definizione 3 (Valore di Page Rank)**.**
Il valore di Page Rank di una determinata pagina P_i, denotato con $r(P_i)$, è definito ricorsivamente come la somma dei PageRanks di tutte le pagine che puntano a P_i, ovvero

$$r_{k+1}(P_i) := \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}. \tag{1.1}$$

Dove B_{P_i} è l'insieme delle pagine che puntano P_i, $|P_j|$ è il numero di outlink dalla pagina P_j e si suppone per ipotesi che tutte le pagine abbiano in partenza lo stesso valore di Page Rank, quindi con $r_0(P_i) = \frac{1}{n}$, dove n = numero di pagine presenti in Google.

Definizione 4 (Vettore Page Rank)**.**
Definiamo il vettore di Page Rank come un vettore colonna $\boldsymbol{\pi} = (r(P_j))_{j=1:n}$ di dimensioni $1 \times n$.

1.3 La matrice di hyperlink H

Definizione 5.
La matrice di hyperlink è una matrice di dimensioni $n \times n$ definita da

$$H_{ij} = \begin{cases} \frac{1}{|P_i|} & \text{se } P_i \text{ punta } P_j \\ 0 & \text{altrimenti} \end{cases}$$

Ad esempio la matrice di hyperlink associata al grafo in fig.1 sarà

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

Osservazione 1.3.1.
Seguono alcune proprietà della matrice H

- H è estremamente **sparsa**, in quanto la maggior parte delle pagine web punta verso poche altre pagine.
- H è **sub-stocastica** per righe, cioè:
  1. $H_{ij} \in \mathbb{R}$ e $H_{ij} \geq 0 \ \forall i, j = 1, 2, \ldots, n$
  2. $\sum_{j=1}^{n} H_{ij} \leq 1 \ \forall i = 1, 2, \ldots, n.$
- H è **riducibile**
  cioè esiste una matrice di permutazione T tale che

  $$TAT^T = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$$

  con X e Z quadrate.
- $\boldsymbol{\pi}$ è un autovettore di H relativo all'autovalore λ=1
  in quanto è possibile riscrivere l'eq.(1.1) sfruttando la seguente forma matriciale

  $$\boldsymbol{\pi} = H^T \boldsymbol{\pi} \tag{1.2}$$

  Oss: Per ogni matrice quadrata H vale che $\boldsymbol{\pi} \in \mathbb{C}$ è autovalore di H se e solo se $\boldsymbol{\pi}$ è autovalore di H^T.

La matrice di hyperlink H presenta, tuttavia, due problemi principali:

1. può presentare righe nulle nel caso ci siano delle pagine che non indirizzano ad altre pagine.

2. non è garantito che H abbia un autovalore 1, e nel caso in cui esista non è detto che sia semplice.

Affrontiamo il primo problema.

1.4 La matrice stocastica S

Definizione 6.
Un nodo j è detto **dangling node** se $|P_j| = 0$.

Nel nostro esempio in fig.1.1 il nodo 2 è un dangling node. Nel Web vi è una presenza massiccia di dangling nodes, infatti immagini, file pdf e tabelle di dati, ad esempio, spesso non indirizzano a nessuna altra pagina.

Per risolvere questo problema innanzitutto definiamo :

Definizione 7 (Vettore dei dangling nodes)**.**

$$\mathbf{d}_j = \begin{cases} 1 & \text{se } |P_j| = 0 \\ 0 & \text{altrimenti} \end{cases}$$

Modifichiamo quindi la matrice di hyperlink H :

$$S := H + \mathbf{d}(\frac{1}{n}\mathbf{e}^T) \tag{1.3}$$

Osservazione 1.4.1.

- S è una matrice **stocastica** per righe, cioè:

  1. $S_{ij} \in \mathbb{R}$ e $S_{ij} \geq 0 \ \forall i, j = 1, 2, \ldots, n$
  2. $\sum_{j=1}^{n} S_{ij} = 1 \ \forall i = 1, 2, \ldots, n.$

  Osserviamo inoltre che in una matrice stocastica l'autovalore più grande in modulo è uguale a 1 (vedi Lemma 4).

- S è riducibile.

- S è sparsa, ma molto meno sparsa di H dato che la maggior parte dei nodi sono dangling nodes.

Rifacendoci al nostro esempio, S sarà dunque :

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

In questo modo il random surfer non si bloccherà più nel nodo 2, ma potrà muoversi in modo equiprobabile verso ogni altra pagina.

1.5 La matrice G di Google

Rimane ancora irrisolto il problema della convergenza al risultato cercato; per questo motivo ci rifaremo al teorema di Perron-Frobenius che assicura l'esistenza e l'unicità di una tale soluzione sotto l'ipotesi di avere una matrice primitiva.

Definizione 8. Una matrice $A \in \mathbb{R}^{n\times n}$ è **primitiva** se e solo se è stocastica e irriducibile.

S è stocastica, quindi dovremo apportare delle ulteriori modifiche per renderla irriducibile.
Vediamo una definizione alternativa a quella già vista di matrice irriducibile.

Definizione 9 (Grafo fortemente connesso)**.**
Il grafo diretto di una matrice A si dice fortemente connesso se per ogni elemento nonzero A_{ij} (coppia di vertici (P_i,P_j)) esiste un percorso nel grafo che porta da P_i a P_j.

Teorema 1. Una matrice A è **irriducibile** se e solo se il suo grafo diretto è fortemente connesso.

Vediamo quindi come aggiornare S in modo che sia anche irriducibile:

Definizione 10 (Parametro di teletrasporto)**.**
Il parametro di teletrasporto è uno scalare $\boldsymbol{\alpha}$ tale che $0 \leq \alpha \leq 1$.
Questo parametro rappresenta la probabilità che un link venga cliccato, pertanto un navigatore del web visiterà una pagina casuale con una probabilità uguale a $(1 - \alpha)$.

Più avanti seguiranno delle discussioni sulla scelta del parametro ottimale α da usare.

Definizione 11.
Fissato α parametro di teletrasporto, definiamo la **matrice Google** come

$$G := \alpha S + (1 - \alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T \tag{1.4}$$

Osservazione 1.5.1.
Proprietà di G:

- G è **stocastica** per righe in quanto combinazione convessa di due matrici stocastiche per righe S e $E = \frac{1}{n}\mathbf{e}\mathbf{e}^T$.
- G è **irriducibile** in quanto ogni pagina è direttamente collegata ad ogni altra pagina.
- G è completamente **densa**. Questo può rappresentare un problema a livello computazionale, tuttavia osserveremo nel Capitolo 3 che G si può riscrivere come un aggiornamento di rango 1 della matrice sparsa H.

Osservazione 1.5.2.
Una modifica alternativa può essere :

$$G = \alpha S + (1-\alpha)\mathbf{e}\mathbf{v}^T, \quad \mathbf{v}^T\mathbf{e} = 1 \tag{1.5}$$

con $\mathbf{v}$ scelto in modo da influenzare la scelta dell'importanza (es. vettore di personalizzazione).

Con queste modifiche, G soddisfa tutte le ipotesi del seguente teorema:

Teorema 2 (Teorema di Perron-Frobenius - variante)**.**
Sia G irriducibile e stocastica per colonna. Allora l'autovalore dominante ($|\lambda_1| = \max|\lambda|$) è uguale ad 1. Inoltre, esiste un unico autovettore associato, π, con componenti tutte positive, e questo è l'unico autovettore con tutte componenti non-negative. Infine, se G ha tutti elementi positivi, allora $|\lambda_i| < 1, i = 2, 3, \ldots, n$.

Dimostrazione. Si veda ([3], pag. 661) □

Siccome G è stocastica e irriducibile, il teorema precedente garantisce che 1 sia autovalore semplice di G e quindi assicura anche l'esistenza ed unicità di $\boldsymbol{\pi}$, suo autovettore associato.

Applichiamo quanto detto finora al piccolo esempio di web in figura 1.1. Troviamo che :

$$G^T\boldsymbol{\pi} = \begin{bmatrix} 0.0250 & 0.4500 & 0.4500 & 0.0250 & 0.0250 & 0.0250 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.3083 & 0.3083 & 0.0250 & 0.0250 & 0.3083 & 0.0250 \\ 0.0250 & 0.0250 & 0.0250 & 0.0250 & 0.4500 & 0.4500 \\ 0.0250 & 0.0250 & 0.0250 & 0.4500 & 0.0250 & 0.4500 \\ 0.0250 & 0.0250 & 0.0250 & 0.8750 & 0.0250 & 0.0250 \end{bmatrix} \boldsymbol{\pi} = \begin{bmatrix} 0.0577 \\ 0.0560 \\ 0.0433 \\ 0.3395 \\ 0.2130 \\ 0.2904 \end{bmatrix}$$

Quindi la pagina corrispondente al nodo numero 4 è la pagina più importante.

Vediamo ora alcune proprietà dello spettro di G:

Definizione 12. Sia $G \in \mathbb{C}^{n\times n}$. I cerchi del piano complesso

$$\mathcal{G}_i^{(r)} := \left\{ z \in \mathbb{C} : |z - \mathbf{A}_{ii}| \leq \sum_{j=1, j\neq i}^{n} |\mathbf{A}_{ij}| \right\}, \quad i = 1, \ldots, n$$

di centro $\mathbf{A}_{ii}$ e raggio $\sum_{j=1,j\neq i}^{n} |\mathbf{A}_{ij}|$ sono detti cerchi di Gerschgorin per righe, da cui l'apice (r).

Lemma 3 (Primo Teorema di Gerschgorin)**.** Sia G una matrice $n \times n$. Ogni autovalore λ di G soddisfa

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

Quindi ogni autovalore di G è contenuto nell'unione dei dischi di Gerschgorin per righe.

(La definizione e il lemma precedenti possono essere trovati in [4])

Lemma 4. Sia G una matrice stocastica $n \times n$. Allora 1 è l'autovalore più grande di G.

Dimostrazione.

$$G\mathbf{e} = G \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} g_{1j} 1 \\ \sum_{i=1}^{n} g_{2j} 1 \\ \vdots \\ \sum_{i=1}^{n} g_{nj} 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = 1\mathbf{e}$$

Pertanto 1 è autovalore di G. Applicando il teorema di Gerschgorin su G, troveremo che per ogni autovalore z di G esiste un qualche $i \in \mathbb{N}$ $1 \leq i \leq n$ tc $|z - g_{ij}| \leq \sum_{j=1,j\neq i}^{n} |g_{ij}|$. Questo porta a

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

Cioè, 1 è l'autovalore più grande di G. □

Teorema 5. Nelle notazioni precedenti, sia spec(S)=$\{1, \lambda_2, \ldots, \lambda_n\}$. Allora spec$(G)$ = $\{1, \alpha\lambda_2, \ldots, \alpha\lambda_n\}$

Dimostrazione.
Sia $\overline{\mathbf{1}} = \frac{1}{\sqrt{n}}\mathbf{1}$.
Innanzi tutto $P^T\overline{\mathbf{1}} = \alpha H^T\overline{\mathbf{1}} + (1-\alpha)\frac{1}{n}\mathbf{1}\mathbf{1}^T\overline{\mathbf{1}} = \overline{\mathbf{1}}$.
Sia ora $U = \left[\overline{\mathbf{1}}, U_0\right]$ unitaria, allora $U^*HU = \begin{bmatrix} 1 & 0 \\ * & T \end{bmatrix}$. Per cui,

$$U^*PU = \alpha \begin{bmatrix} 1 & 0 \\ * & T \end{bmatrix} + (1-\alpha)\frac{1}{n} n\mathbf{e}_1\overline{\mathbf{1}}^T U = \begin{bmatrix} 1 & 0 \\ * & \alpha T \end{bmatrix}$$

. □

La precedente dimostrazione è riportata in [5] .

Osservazione 1.5.3.

- Anche se 1 è autovalore multiplo di H, è semplice per G.
- Siano $\tilde{\lambda_1}, \tilde{\lambda_2}, \ldots, \tilde{\lambda_n}$ gli autovalori di G ordinati in modo decrescente.
  Deduciamo dal lemma precedente che $\left|\tilde{\lambda_1}\right| = 1$ e $\left|\tilde{\lambda_i}\right| \leq \alpha \ \forall i \geq 2$.

Il teorema seguente ci permette di concludere che, se G soddisfa una determinata ipotesi, il secondo autovalore è esattamente uguale ad α.

Teorema 6. Sia S una matrice $n \times n$ stocastica per righe. Sia α un numero reale tale che $0 \leq \alpha \leq 1$. Sia $E = \mathbf{e}\mathbf{v}^T$ una matrice stocastica per righe di rango 1, dove $\mathbf{v}$ è un vettore con n componenti che rappresenta una distribuzione di probabilità. Definita la matrice $G = \alpha S + (1 - \alpha)E$, il suo secondo autovalore soddisfa $|\lambda_2| \leq \alpha$.
Inoltre se S ha almeno due sottoinsiemi chiusi irriducibili (ed è il caso della matrice di hyperlink del web), allora il secondo autovalore di G soddisfa $|\lambda_2| = \alpha$.

Dimostrazione. si veda [6] □

Osserviamo che scegliendo $v = \frac{1}{n}\mathbf{e}$ questo teorema risulta particolarmente appropriato.

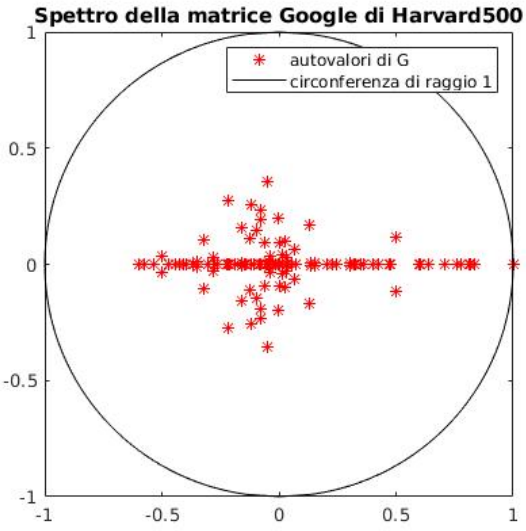


Figura 1.2: Spettro di una matrice Google

Capitolo 2

Calcolo del Page Rank come problema agli autovalori

Nel primo capitolo abbiamo dimostrato che il vettore di PageRank è l'unico vettore di norma unitaria che soddisfa $\boldsymbol{\pi} = G^T\boldsymbol{\pi}$, pertanto il nostro obiettivo è trovare l'autocoppia dominante della matrice G.
Essendo il World Wide Web formato da miliardi di pagine, questo problema non è affatto banale. Per tale motivo nei prossimi capitoli studieremo vari metodi che permetteranno di calcolare questa autocoppia nel modo più veloce possibile.

2.1 Il metodo delle potenze

Nonostante nel 1996 fossero presenti già numerosi algoritmi efficienti nella ricerca dell'autocoppia dominante, nessuno di questi era in grado di far fronte alle grandi dimensioni di questo nuovo problema. Per questo motivo bisognò ripartire dalle origini, ovvero dal metodo delle potenze.
Il metodo delle potenze è un metodo iterativo finalizzato al calcolo dell'autovalore dominante di una matrice $A \in \mathbb{R}^{n\times n}$ ed il relativo autovettore associato.

2.1.1 L'algoritmo

Algorithm 1: Metodo delle Potenze

Dati $A \in \mathbb{R}^{n\times n}$ diagonalizz., $x^{(0)} \in \mathbb{C}^n$ di norma unitaria, $y = Ax^{(0)}$;
for *i=0,1,2,...,fino a convergenza* **do**
  $\lambda^{(i)} = (x^{(i)})^H y$;
  if $\|x^{(i)} - x^{(i-1)}\|_1 < \epsilon$ **then**
    return
  else
    $x^{(i+1)} = y/\|y\|$;
    $y = Ax^{(i+1)}$
  end
end

Notiamo che $\lambda^{(i)}$ si ottiene applicando il quoziente di Rayleigh, ricordando che il vettore $x^{(i)}$ ha norma unitaria.
Indicata con (λ_1, x_1) l'autocoppia dominante di A, il metodo delle potenze costruisce due successioni $\{x^{(k)}\}_{k\in\mathbb{N}}$ e $\{\lambda^{(k)}\}_{k\in\mathbb{N}}$ tali che

$$x^{(k)} \to x_1, \quad \lambda^{(k)} \to \lambda_1, \quad k \to +\infty$$

2.1.2 Convergenza

Teorema 7. Se A è diagonalizzabile, $x^{(0)}$ è di norma unitaria e l'autovalore dominante λ_1 è semplice in modulo, allora il metodo delle potenze converge sempre all'autocoppia dominante, .

Dimostrazione. A è per ipotesi diagonalizzabile, pertanto esiste una base di $\mathbb{R}^n$ di autovettori $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Allora possiamo scrivere $x^{(0)}$ come combinazione lineare di questi autovettori per certi coefficienti $c_1, \ldots, c_n \in \mathbb{R}$.

$$x_0 = \sum_{i=1}^{n} c_i \mathbf{v}_i$$

Ora, applicando l'iterazione descritta dal metodo delle potenze otteniamo che :

$$x^{(1)} = Ax^{(0)} = A\sum_{i=1}^{n} c_i \mathbf{v}_i = \sum_{i=1}^{n} c_i A\mathbf{v}_i = \sum_{i=1}^{n} c_i \lambda_i \mathbf{v}_i$$

$$\vdots$$

$$x^{(k)} = Ax^{(k-1)} = A\sum_{i=1}^{n} c_i \lambda_i^{k-1} \mathbf{v}_i = \sum_{i=1}^{n} c_i \lambda_i^{k-1} A\mathbf{v}_i = \sum_{i=1}^{n} c_i \lambda_i^{k} \mathbf{v}_i.$$

Dividiamo tutto per $(\lambda_1)^k$, ottenendo così

$$\frac{x^{(k)}}{\lambda_1^k} = c_1\mathbf{v}_1 + \sum_{i=2}^{n} c_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \mathbf{v}_i.$$

Siccome $|\lambda_i| \leq |\lambda_1| \ \forall i \geq 2$, mandando $k \to \infty$ otteniamo che le iterate $x^{(k)}$ tendono alla direzione dell'autovettore $\mathbf{v}_1$:

$$\lim_{k\to\infty} \frac{x^{(k)}}{\lambda_1} = c_1 v_1.$$

□

Abbiamo così dimostrato anche il seguente risultato:

Teorema 8. Siano $|\lambda_i|$, $i = 1, \ldots, n$ ordinati in modo decrescente e $|\lambda_1| > |\lambda_2|$, con $|\lambda_1|$ semplice. Sia $c_1 \neq 0$, allora esiste una costante $\mathrm{C} > 0$, tale che

$$\|x^{(k)} - x_1\|_2 \leq C \left|\frac{\lambda_2}{\lambda_1}\right|^k, k \geq 1.$$

Osservazione 2.1.1. Il metodo converge tanto più velocemente, quanto la differenza tra $|\lambda_1|$ e $|\lambda_2|$ è grande.

2.1.3 Applicazione al Page Rank

Ai fini di calcolare il vettore di PageRank $\boldsymbol{\pi}$ applichiamo l'algoritmo delle Potenze alla matrice G^T, scegliendo come iterata iniziale $x^{(0)} = \frac{1}{n}\mathbf{e}$. Come già notato in precedenza la matrice G non è sparsa, quindi sarà preferibile riscriverla come nell'equazione 1.5, ottenendo

$$G = \alpha S + (1-\alpha)\mathbf{e}\mathbf{v}^T = \alpha\left(H + \mathbf{d}\mathbf{v}^T\right) + (1-\alpha)\,\mathbf{e}\mathbf{v}^T.$$

Per quanto riguarda la convergenza, dal teorema 8 segue che l'ordine di convergenza del metodo sarà $\mathcal{O}(\alpha^k)$. E' immediato vedere che per valori piccoli di α il metodo convergerà molto velocemente, mentre per valori di α che si avvicinano ad 1 si darà priorità alla struttura naturale di hyperlink del Web a discapito della velocità di convergenza. Brin e Page scelsero come valore ottimale $\alpha = 0.85$, un parametro che trova un compromesso tra efficacia ed efficienza. Vediamo come varia la convergenza del metodo al variare del parametro α.

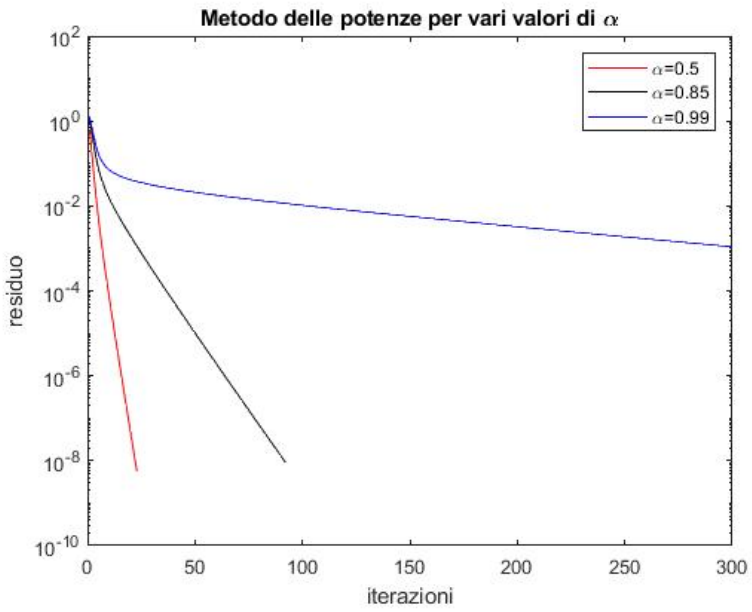


Figura 2.1: Confronto della convergenza del metodo delle potenze per vari valori di α

Il seguente grafico è stato ottenuto usando come matrice di hyperlink la matrice Stanford-Berkley web (vedi capitolo 4), maxit=300, tolleranza=1e-8 e vettore di personalizzazione $\mathbf{v}=\mathbf{e}/\mathrm{n}$.

2.2 Ottimizzazione del metodo delle potenze

Un vantaggio del metodo delle potenze è il fatto che si possa scegliere un vettore iniziale; chiaramente più questo viene scelto vicino al vettore di PageRank, meno iterazioni serviranno per raggiungere un'approssimazione soddisfacente di $\boldsymbol{\pi}$. Sfruttando questo criterio vediamo ora una modifica del metodo delle potenze presentato prima.

2.2.1 Vettore di PageRank atteso

Abbiamo visto che il vettore di PageRank dipende fortemente dal valore di α, pertanto per ogni α potremo calcolare $\boldsymbol{\pi}(\alpha)$. Supponiamo ora di voler trovare un unico vettore di PageRank che riassuma tutti i valori che $\boldsymbol{\pi}(\alpha)$ può assumere al variare di α.

Definizione 13.
Sia $f : [0,1] \to \mathbb{R}^+$ la funzione di densità di probabilità del parametro stocastico α. Definiamo **il vettore di PageRank atteso** $\langle\boldsymbol{\pi}\rangle$ come

$$\langle\boldsymbol{\pi}\rangle = \int_0^1 \boldsymbol{\pi}(\alpha) f(\alpha) d\alpha.$$

Tuttavia per funzioni di densità f generiche questa formula non risulta essere estremamente comoda per calcolare $\langle\boldsymbol{\pi}\rangle$, pertanto cercheremo di approssimarlo. Scegliamo

dunque una partizione dell'intervallo chiuso $[0,1]$ del tipo $0 = x_0 < x_1 < \cdots < x_k = 1$ e selezioniamo un elemento in ogni intervallino $\alpha_i \in [x_{i-1}, x_i]$ per ogni $1 \leq i \leq k$, $i \in \mathbb{N}$. Possiamo approssimare il vettore $\langle \boldsymbol{\pi} \rangle$ attraverso una somma di Riemann :

$$\langle \boldsymbol{\pi} \rangle \approx \sum_{i=1}^{k} \pi(\alpha_i) f(\alpha_i) \left[x_i - x_{i-1} \right].$$

Segue banalmente che più la partizione è fitta, cioè più k è grande, più l'approssimazione sarà precisa.

2.2.2 L'algoritmo

Approssimare il vettore di PageRank atteso $\langle \boldsymbol{\pi} \rangle$ richiede il calcolo di $\boldsymbol{\pi}(\alpha)$ per vari valori di α, pertanto questo processo potrebbe sembrare molto dispendioso. Tuttavia, come già accennato a inizio sezione, il metodo delle potenze ha il vantaggio di richiedere un vettore iniziale, pertanto sfrutteremo questa proprietà prendendo come vettore iniziale il vettore di PageRank relativo ad un α diverso da quello ottimale. Possiamo applicare dunque il seguente algoritmo:

Algorithm 2: Metodo delle Potenze ottimizzato

$\boldsymbol{\pi}(\alpha_1) = potenze(\mathbf{v})$;
for *k=2:m* **do**
  $\boldsymbol{\pi}(\alpha_k) = potenze(\boldsymbol{\pi}(\alpha_{k-1}))$;
end

dove con "$potenze(\mathbf{w})$" indichiamo l'applicazione del metodo delle potenze che utilizzi $\mathbf{w}$ come vettore iniziale.
Questo algoritmo sfrutta l'idea che se due valori α e α' sono vicini allora anche $\boldsymbol{\pi}(\alpha)$ e $\boldsymbol{\pi}(\alpha')$ lo sono. In generale questa affermazione è vera per valori di α lontani da 1, tuttavia notiamo che

$$\boldsymbol{\pi} = (1-\alpha)\left(I - \alpha S^T\right)^{-1} \mathbf{v}$$

Il più piccolo autovalore di $\left(I - \alpha S^T\right)$ è esattamente $1-\alpha$ e ciò vuol dire che il numero di condizionamento di questa matrice vale $\frac{1+\alpha}{1-\alpha}$. Notiamo che se α tende a 1 il numero di condizionamento tende a infinito, pertanto una piccola variazione della matrice $\left(I - \alpha S^T\right)$, data ad esempio da una variazione di α, potrebbe produrre un grande cambiamento di $\left(I - \alpha S^T\right)^{-1}$ e di conseguenza il vettore di PageRank potrebbe cambiare notevolmente per due valori $\alpha \approx \alpha'$ vicini a 1.
Per questo motivo nella sezione 3.2 studieremo un metodo molto più efficiente in grado di approssimare il vettore di PageRank per tutti i valori di α.

2.3 Algoritmo di Golub e Greif

L'ultimo metodo che tratteremo, e che vede il calcolo del Page Rank come un problema agli autovalori, è il metodo presentato da Golub e Greif in [7]. Non entreremo nel dettaglio, ma accenniamo che appartiene alla classe dei metodi basati sull'algoritmo di Arnoldi (in dettaglio nella sezione 3.2) e che fa uso della scomposizione in valori singolari.

Capitolo 3

Approssimazione del vettore di PageRank mediante risoluzione di un sistema lineare

Inizialmente Brin e Page concepirono il problema del Page Rank come un problema agli autovalori, ma successivamente esso acquisì un "secondo volto": quello di sistema lineare.
Come già accennato nell'Osservazione 1.5.1, vediamo che la matrice densa G si può riscrivere come un aggiornamento di rango 1 della matrice sparsa H nel seguente modo:

$$\begin{aligned} G &= \alpha S + (1-\alpha)\mathbf{e}\mathbf{v}^T \\ &= \alpha(H + \mathbf{d}\mathbf{v}^T) + (1-\alpha)\mathbf{e}\mathbf{v}^T \\ &= \alpha H + (\alpha\mathbf{d} + (1-\alpha)\mathbf{e})\mathbf{v}^T. \end{aligned} \tag{3.1}$$

Pertanto, ponendo $\boldsymbol{\pi}^T\mathbf{d} = \gamma$ e ricordando che $\boldsymbol{\pi}^T\mathbf{e} = 1$, otteniamo che il problema agli autovalori $\boldsymbol{\pi}^T = G\boldsymbol{\pi}^T$ si trasforma con dei passaggi algebrici nel seguente sistema lineare

$$\boldsymbol{\pi}^T (I - \alpha H) = (1 - \alpha + \alpha\gamma)\, \mathbf{v}^T. \tag{3.2}$$

Siccome al termine del metodo normalizzeremo $\boldsymbol{\pi}$, possiamo scegliere arbitrariamente un valore conveniente di γ, ad esempio γ=1.
In definitiva otteniamo la seguente equazione che mostra in che modo il problema del PageRank possa essere visto come risoluzione di un sistema lineare

$$\boldsymbol{\pi}^T (I - \alpha H) = \mathbf{v}^T \tag{3.3}$$

che è equivalente a $\left(I - \alpha H^T\right)\boldsymbol{\pi} = \mathbf{v}$.
Molti ricercatori hanno affrontato il problema in questo nuovo modo. Bianchini e altri hanno suggerito l'utilizzo del metodo di Jacobi per calcolare il vettore di Page Rank [8], mentre nei nostri esperimenti numerici utilizzeremo la funzione Matlab `bicgstabl`, ovvero il metodo del bi-gradiente coniugato stabilizzato, e un nuovo algoritmo che tratteremo nella sezione 3.2.

3.1 Il metodo di Jacobi

Definizione 14.
Una matrice A di ordine n si dice matrice a dominanza diagonale stretta per righe se

$$|\mathbf{A}_{ii}| > \sum_{j=1, j\neq i}^{n} |\mathbf{A}_{ij}|, \quad i = 1, \ldots, n.$$

Ricordiamo che il nostro obiettivo è trovare una soluzione del sistema lineare

$$\left(I - \alpha H^T\right) \mathbf{x} = \mathbf{.}$$

La matrice $\left(I - \alpha H^T\right)$ è a dominanza diagonale stretta per righe, pertanto la convergenza del metodo di Jacobi applicato a questa matrice è garantita (si veda [4], pag. 40).

Inoltre si osservi che

$$\left(I - \alpha H^T\right) \mathbf{x} = \mathbf{v} \iff \mathbf{x} - \alpha H^T \mathbf{x} = \mathbf{v} \iff \mathbf{x} = \alpha H^T \mathbf{x} + \mathbf{v}$$

per questo motivo segue il seguente algoritmo:

Algorithm 3: Metodo di Jacobi applicato alla matrice αH^T

Sia $k = 0$; $x^{(0)} = \mathbf{v}$;
for *k da 1 a convergenza* **do**
  $x^{(k+1)} = \alpha H^T x^{(k)} + \mathbf{v}$
end

La dimostrazione che questo metodo converga al vettore $\boldsymbol{\pi}$ si può trovare in ([9], pag.23).

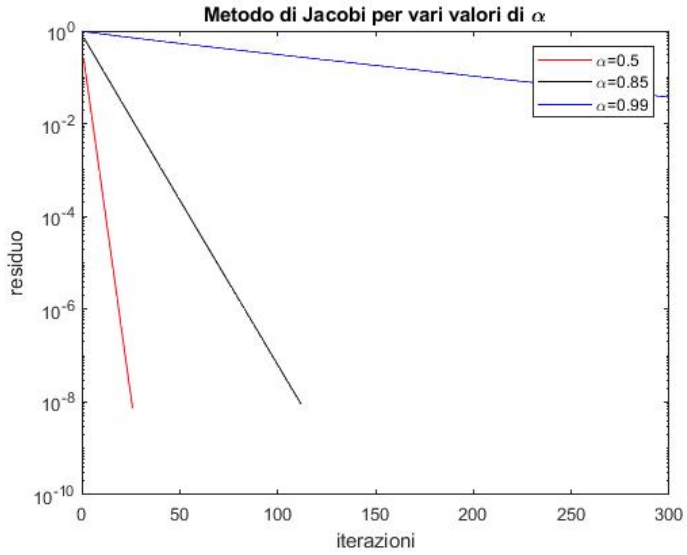


Figura 3.1: Confronto della convergenza del metodo di Jacobi per vari valori di α della matrice di hyperlink H_{SB} (vedi capitolo 4).

3.2 Algoritmo di Arnoldi e modello di ordine ridotto

D'ora in poi denoteremo con $\boldsymbol{\pi}(\alpha)$ il vettore di PageRank $\boldsymbol{\pi}$, sottolineando così la dipendenza di $\boldsymbol{\pi}$ dal valore del parametro di teletrasporto α. Il nostro obiettivo è quello di approssimare $\boldsymbol{\pi}(\alpha)$ per diversi valori di α per poi poter calcolare il vettore di PageRank atteso $\langle\boldsymbol{\pi}\rangle$ introdotto nella sezione 2.2.1.
Dalla (3.2) sappiamo che il vettore di PageRank corrispondente ad un certo valore di α è la soluzione del sistema lineare $\mathbf{x}(\alpha)^T (I - \alpha H) = (1 - \alpha + \alpha\gamma)\, \mathbf{v}^T$. Ad inizio capitolo abbiamo però mostrato che è sufficiente risolvere il sistema $\left(I - \alpha H^T\right)\mathbf{x}(\alpha) = \mathbf{v}$, pertanto ci limiteremo a risolvere quest'ultimo e per farlo sfrutteremo l'invarianza per traslazione degli spazi di Krylov.

Definizione 15 (Spazio di Krylov)**.** Sia $A \in \mathbb{R}^{l\times l}$ e $\mathbf{w}\in \mathbb{R}^l$, definiamo lo spazio di Krylov K_m come

$$K_m(A, \mathbf{w}) = span\{\mathbf{w}, A\mathbf{w}, A^2\mathbf{w}, \cdots, A^{m-1}\mathbf{w}\}$$

Osservazione 3.2.1 (Alcune proprietà degli spazi di Krylov)**.**

- Gli spazi di Krylov sono **invarianti per traslazione**, cioè dati σ e τ due scalari nelle notazioni precedenti si ha che $K_m(A + \sigma I, \mathbf{w}) = K_m(A + \tau I, \mathbf{w})$.
- Uno spazio di Krylov $K_m(A, \mathbf{w})$ è uno spazio vettoriale, dunque si ha che $K_m(A, \mathbf{w}) = K_m(\beta A, \mathbf{w})$ per ogni $\beta \neq 0$.

Allora in particolare si ha che $K_m(A, \mathbf{w}) = K_m(I - \beta A, \mathbf{w})\ \forall\beta \neq 0$.

Concentriamoci ora sullo spazio di Krylov $K_n(H^T, \mathbf{v})$, cioè lo spazio di Krylov della matrice H^T rispetto al vettore di personalizzazione $\mathbf{v}$.
Il seguente teorema è molto importante in quanto conferma che la soluzione dell'equazione $\left(I - \alpha H^T\right)\mathbf{x} = \mathbf{v}$ appartenga effettivamente a $K_n\left(H^T, \mathbf{v}\right)$ e garantisce la convergenza dell'algoritmo.

Teorema 9. Il vettore di PageRank $\boldsymbol{\pi}$ appartiene allo spazio di Krylov $K_n(H^T, \mathbf{v})$.

Dimostrazione. Notiamo innanzitutto che $K_m(H^T, \mathbf{v}) = K_m(I - \alpha H^T, \mathbf{v})$. Sia $p(\lambda) = c_0 + c_1\lambda + c_2\lambda^2 + \cdots + \lambda^n$ il polinomio caratteristico di $(I - \alpha H^T)$. Per il teorema di Cayley-Hamilton $I - \alpha H^T$ soddisfa

$$0 = p\left(I - \alpha H^T\right) = c_0 I + c_1\left(I - \alpha H^T\right) + c_2\left(I - \alpha H^T\right)^2 + \cdots + \left(I - \alpha H^T\right)^n$$

Siccome $\left(I - \alpha H^T\right)$ è invertibile, 0 non è un autovalore, pertanto $c_0 \neq 0$. Dividendo per c_0, sottraendo I e infine moltiplicando per $\left(I - \alpha H^T\right)^{-1}$ si ottiene

$$\left(I - \alpha H^T\right)^{-1} = -\frac{1}{c_0}\left(c_1 + c_2\left(I - \alpha H^T\right) + \cdots + (I - \alpha H^T)^{n-1}\right)$$

Da cui,

$$\mathbf{x} = \left(I - \alpha H^T\right)^{-1}\mathbf{v} = -\frac{1}{c_0}\left(c_1\mathbf{v} + c_2\left(I - \alpha H^T\right)\mathbf{v} + \cdots + \left(I - \alpha H^T\right)^{n-1}\mathbf{v}\right)$$

ed è evidente che l'ultimo termine $\in K_n\left(I - \alpha H^T, \mathbf{v}\right)$.
Pertanto $\boldsymbol{\pi} = \frac{\mathbf{x}}{e^T\mathbf{x}} \in K_n\left(I - \alpha H^T, \mathbf{v}\right) = K_n(H^T, \mathbf{v})$. □

Una volta individuato lo spazio dove dovremo cercare la nostra soluzione $\boldsymbol{\pi}$, abbiamo bisogno di più informazioni su di esso per poterlo usare, per questo ricorriamo all'algoritmo di Arnoldi, un algoritmo simile al già noto algoritmo di Gram-Schmidt in grado di fornire una base ortonormale di $K_n\left(H^T, \mathbf{v}\right)$ se applicato ai vettori $\mathbf{v}$,$H^T\mathbf{v}$,$(H^T)^2\mathbf{v}$,$\cdots$,$(H^T)^{(n-1)}\mathbf{v}$.

Algorithm 4: Algoritmo di Arnoldi

Sia $\mathbf{w}_1 = \mathbf{v}/\|\mathbf{v}\|_2$;
for *k=1:m* **do**
  $\mathbf{w}_{k+1} = H^T\mathbf{w}_k$ $n(2n-1)$ flops;
  for *j=1:k* **do**
    $u_{j,k} = \langle\mathbf{w}_j, \mathbf{w}_{k+1}\rangle$ $2n-1$ flops;
    $\mathbf{w}_{k+1} = \mathbf{w}_{k+1} - u_{j,k}\mathbf{w}_j$ $2n$ flops
  end
  $u_{k+1,k} = \|\mathbf{w}_{k+1}\|_2$ $2n-1$ flops;
  if $u_{k+1,k} = 0$ **then**
    stop
  else
    $\mathbf{w}_{k+1} = \mathbf{w}_{k+1}/u_{k+1,k}$ n flops
  end
end

Teorema 10. Supponendo che l'algoritmo di Arnoldi non si interrompa prima di m iterazioni, esso crea una base ortonormale dello spazio $K_m(H^T, \mathbf{v})$.

Dimostrazione. Mostriamo per induzione che $\mathbf{w}_k = p_{k-1}\left(H^T\right)\mathbf{v}$ per un qualche polinomio p_{k-1} di grado $k-1$. Il caso $k = 1$ è banale in quanto supponiamo per ipotesi $\mathbf{w}_1 = \mathbf{v}/\|\mathbf{v}\|_2$. Consideriamo ora $k \in \mathbb{N}$, $k \geq 2$. Notiamo che

$$\begin{aligned}\mathbf{w}_{k+1} =& \frac{1}{u_{k+1,k}}\left(H^T\mathbf{w}_k - \sum_{j=1}^{k} u_{j,k}\mathbf{w}_j\right)\\ =& \frac{1}{u_{k+1}}\left(H^T p_{k-1}\left(H^T\right)\mathbf{v} - \sum_{j=1}^{k} u_{j,k}p_{j-1}\left(H^T\right)\mathbf{v}\right).\end{aligned}$$

Questo mostra che $\mathbf{w}_{k+1} = p_k\left(H^T\right)\mathbf{v}$, dove il polinomio p_k è definito come $p_k(x) = (xp_{k-1}(x) - \sum_{j=1}^{k} u_{j,k}p_{j-1}(x))/u_{k+1,k})$. Siccome abbiamo supposto che p_{k-1} abbia grado $k-1$, il grado di p_k è esattamente k. Inoltre, i vettori generati dall'algoritmo di Arnoldi

sono ortonormali per costruzione quindi formano una base ortonormale di $K_m\left(H^T, \mathbf{v}\right)$ $\square$

Osservazione 3.2.2.
Sia W la matrice $n \times m$ in cui la i-esima colonna è data dall'i-esimo vettore $\mathbf{w}_i$ generato dall'algoritmo di Arnoldi e sia U la matrice $m \times m$ formata dai coefficienti $u_{i,j}$. Notiamo che la matrice U è una matrice di Hessenberg superiore, cioè $U_{i,j} = 0 \ \forall i > j + 1$.
Con queste notazioni è possibile riscrivere l'algoritmo di Arnoldi in forma matriciale con una singola equazione come:

$$H^T W = WU + u_{m+1,m}\mathbf{w}_{m+1}\mathbf{e}_m^T,$$

dove $\mathbf{e}_m$ indica il vettore $(0, \cdots, 0, 1)^T$ di dimensione $m \times 1$.

Giunti a questo punto è lecito chiedersi che cosa possa succedere se l'algoritmo di Arnoldi si dovesse interrompere prima di compiere m iterazioni, ovvero prima di aver creato tutti gli m vettori richiesti.
Il seguente teorema mostra come questo fatto possa essere in effetti una cosa vantaggiosa.

Teorema 11. Supponiamo che l'algoritmo di Arnoldi si interrompa dopo $k < m$ iterazioni. Allora la dimensione dello spazio di Krylov è massima, cioè $K_k(H^T, \mathbf{v}) = K_l(H^T, \mathbf{v})$ $\forall l \geq k$.

Dimostrazione. Se l'algoritmo di Arnoldi si interrompe alla k-esima iterata vuol dire che $u_{k+1,k} = 0$ e quindi $\mathbf{w}_{k+1} = 0$. Pertanto si avrà anche

$$H^T \mathbf{w}_k = \sum_{j=1}^{k} u_{j,k}\mathbf{w}_j.$$

Sia ora $\mathbf{z}$ un elemento arbitrario appartenente a $K_k\left(H^T, \mathbf{v}\right)$. Dal teorema precedente sappiamo che possiamo scrivere $\mathbf{z}$ come combinazione lineare dei vettori di Arnoldi: $\mathbf{z} = \sum_{j=1}^{k} c_j \mathbf{w}_j$ per certi coefficienti c_i. Segue che

$$\begin{aligned} H^T\mathbf{z} =& H^T \sum_{j=1}^{k} c_j \mathbf{w}_j = \sum_{j=1}^{k} c_j H^T \mathbf{w}_j = \sum_{j=1}^{k-1} c_j H^T \mathbf{w}_j + c_k H^T \mathbf{w}_k \\ =& \sum_{j=1}^{k-1} c_j H^T \mathbf{w}_j + c_k \sum_{j=1}^{k} u_{j,k}\mathbf{w}_j \end{aligned}$$

che è un elemento di $K_k\left(H^T, \mathbf{v}\right)$. Quindi $K_{k+1}\left(H^T, \mathbf{v}\right) \subseteq K_k\left(H^T, \mathbf{v}\right)$. L'inclusione inversa è banale in quanto $K_{k+1}\left(H^T, \mathbf{v}\right) = K_k\left(H^T, \mathbf{v}\right)$. La tesi segue dunque per induzione. $\square$

Riassumendo abbiamo innanzitutto dimostrato che il vettore di PageRank $\boldsymbol{\pi}$ appartiene allo spazio $K_n(H^T, \mathbf{v})$. Successivamente, dopo aver analizzato come l'algoritmo di

Arnoldi sia in grado di fornire una base ortonormale di questo spazio, abbiamo notato che se questo algoritmo si dovesse interrompere precocemente fornirebbe una base di uno spazio di Krylov di dimensione apparentemente inferiore $K_k(H^T, \mathbf{v})$. Tuttavia grazie al Teorema 11 abbiamo dimostrato che lo spazio $K_k(H^T, \mathbf{v})$ è equivalente ad ogni $K_l(H^T, \mathbf{v})$ $\forall l \geq k$, pertanto per il Teorema 9 il vettore cercato $\boldsymbol{\pi}$ appartiene anche allo spazio $K_m(H^T, \mathbf{v})$ per $m < n$, senza bisogno di aumentare lo spazio ottenuto: si parla infatti di "happy breakdown".
Ricordiamo che stiamo cercando un vettore $\mathbf{u} \in K_m(H^T, \mathbf{v})$ che approssimi $\left(I - \alpha H^T\right)\mathbf{u} \approx \mathbf{v}$. Siccome $\mathbf{u} \in K_m(H^T, \mathbf{v})$ possiamo scrivere $\mathbf{u}$ come combinazione lineare degli m vettori generati da Arnoldi, cioè $\mathbf{u} = a_1\mathbf{w}_1 + a_2\mathbf{w}_2 + \cdots + a_m\mathbf{w}_m$ o più brevemente $\mathbf{u} = W\mathbf{a}$ con $\mathbf{a} = [a_1, \cdots, a_m]^T$. Il nostro obiettivo è dunque quello di trovare dei buoni coefficienti a_i in modo che la norma del residuo sia sufficientemente bassa.
Analizziamo il **residuo**.
Per definizione il residuo $\mathbf{r}$ è dato da

$$\mathbf{r} = \mathbf{v} - (I - \alpha H^T)\mathbf{u} = \mathbf{v} - \mathbf{u} + \alpha H^T\mathbf{u}$$

Siccome $\mathbf{v} \in K_m(H^T, \mathbf{v})$ e $\mathbf{u} \in K_m(H^T, \mathbf{v})$ segue che $\mathbf{r} \in K_{m+1}(H^T, \mathbf{v})$, dunque è possibile scriverlo come $\mathbf{r} = b_1\mathbf{w}_1 + \cdots + b_{m+1}\mathbf{w}_{m+1}$ per certi coefficienti b_i. Il teorema seguente fornisce un metodo alternativo per ottenere una soluzione nello spazio $K_m(H^T, \mathbf{v})$.

Teorema 12. Nelle notazioni precedenti, scegliendo $\mathbf{a}(\alpha) = \|\mathbf{v}\|_2 \left(I - \alpha U\right)^{-1} \mathbf{e}_1$ si ottiene che $b_i = 0$ $\forall i \in \{1, 2, \ldots, m\}$

Dimostrazione. Sapendo che $\mathbf{u} = W\mathbf{a}(\alpha)$ e $H^TW = WU + u_{m+1,m}\mathbf{w}_{m+1}\mathbf{e}_m^T$ per l'oss. 3.2.2 abbiamo che

$$\begin{aligned}
\mathbf{r} =& \mathbf{v} - (I - \alpha H^T)\mathbf{u} = \mathbf{v} - (I - \alpha H^T)W\mathbf{a}(\alpha) = \\
=& \mathbf{v} - W\mathbf{a}(\alpha) + \alpha H^T W\mathbf{a}(\alpha) = \mathbf{v} - W\mathbf{a}(\alpha) + \alpha\left(WU + u_{m+1,m}\mathbf{u}_{m+1}\mathbf{e}_m^T\right)\mathbf{a}(\alpha) \\
=& \mathbf{v} - W\left(I - \alpha U\right)\mathbf{a}(\alpha) + \alpha u_{m+1,m}\mathbf{w}_{m+1}\left(\mathbf{e}_m^T\mathbf{a}(\alpha)\right).
\end{aligned}$$

Siccome $\mathbf{w}_1 = \mathbf{v}/\|\mathbf{v}\|_2$, abbiamo che $\|\mathbf{v}\|_2 W\mathbf{e}_1 = \mathbf{v}$. Quindi

$$\mathbf{r} = W\left[\|\mathbf{v}\|_2\mathbf{e}_1 - \left(I - \alpha U\right)\mathbf{a}(\alpha)\right] + \alpha u_{m+1,m}\left(\mathbf{e}_m^T\mathbf{a}(\alpha)\right)\mathbf{w}_{m+1}.$$

Usando l'ipotesi che $\mathbf{a}(\alpha) = \|\mathbf{v}\|_2\left(I - \alpha U\right)^{-1}\mathbf{e}_1$, troviamo

$$\mathbf{r} = \alpha u_{m+1,m}\mathbf{e}_m^T\left(I - \alpha U\right)^{-1}\mathbf{e}_1\mathbf{w}_{m+1}.$$

Quindi abbiamo riscritto $\mathbf{r}$ come $\mathbf{r} = b_{m+1}\mathbf{w}_{m+1}$,
con $b_{m+1} = \alpha u_{m+1,m}\mathbf{e}_m^T\left(I - \alpha U\right)^{-1}\mathbf{e}_1 = \alpha u_{m+1,m}c(\alpha)$ dove $c(\alpha)$ indica il primo elemento dell'ultima riga di $\left(I - \alpha U\right)^{-1}$. □

Osservazione 3.2.3.

- abbiamo sottolineato la dipendenza di $\mathbf{a}$ da α, scrivendo $\mathbf{a}(\alpha)$.
- Il principale vantaggio di questo metodo è che, scegliendo $\mathbf{a}(\alpha) = \|\mathbf{v}\|_2 (I - \alpha U)^{-1} \mathbf{e}_1$, è richiesto solamente il calcolo dell'inversa della matrice $(I - \alpha U)^{-1}$ che ha dimensioni $m \times m$. Siccome in genere $m \ll n$ invertire una matrice di dimensione m è molto più veloce di calcolare un prodotto matrice-vettore di dimensione n. Per questo motivo questo metodo è chiamato di "ordine ridotto".

Vediamo dunque l'algoritmo completo:

Algorithm 5:

---

1. fissare un m abbastanza grande;
2. applicare l'algoritmo di Arnoldi a H^T e $\mathbf{v}$ per m iterazioni in modo da calcolare la matrice con colonne ortonormali W e la Hessenberg superiore U;
3. per ogni valore di α, approssimare $\mathbf{x}(\alpha)$ attraverso
$$\mathbf{x}(\alpha) \approx W\mathbf{a}(\alpha) = W\|\mathbf{v}\|_2 (I - \alpha U)^{-1} \mathbf{e}_1;$$
4. normalizzare il vettore trovato per avere una approssimazione del vettore di PageRank
$$\boldsymbol{\pi}(\alpha) = \frac{\mathbf{x}(\alpha)}{\mathbf{e}^T \mathbf{x}(\alpha)}.$$

Una cosa importante da notare è che l'algoritmo di Arnoldi è indipendente dal valore di α, pertanto può essere applicato un'unica volta all'inizio. Fatto questo, l'algoritmo ci permette di approssimare in maniera molto efficiente $\boldsymbol{\pi}(\alpha)$ per vari valori di α.
Infine quanto deve essere preso grande m nel primo passaggio? Un buon modo per deciderlo è creare uno spazio di Krylov $K_m(H^T, \mathbf{v})$ per un certo valore m iniziale usando l'algoritmo di Arnoldi e applicare successivamente l'algoritmo 5. A questo punto se il residuo ottenuto è più grande di quello cercato questo vorrà dire che m è troppo piccolo, dunque si può espandere lo spazio $K_m(H^T, \mathbf{v})$ continuando l'algoritmo di Arnoldi finché si ottiene un residuo dell'ordine cercato.

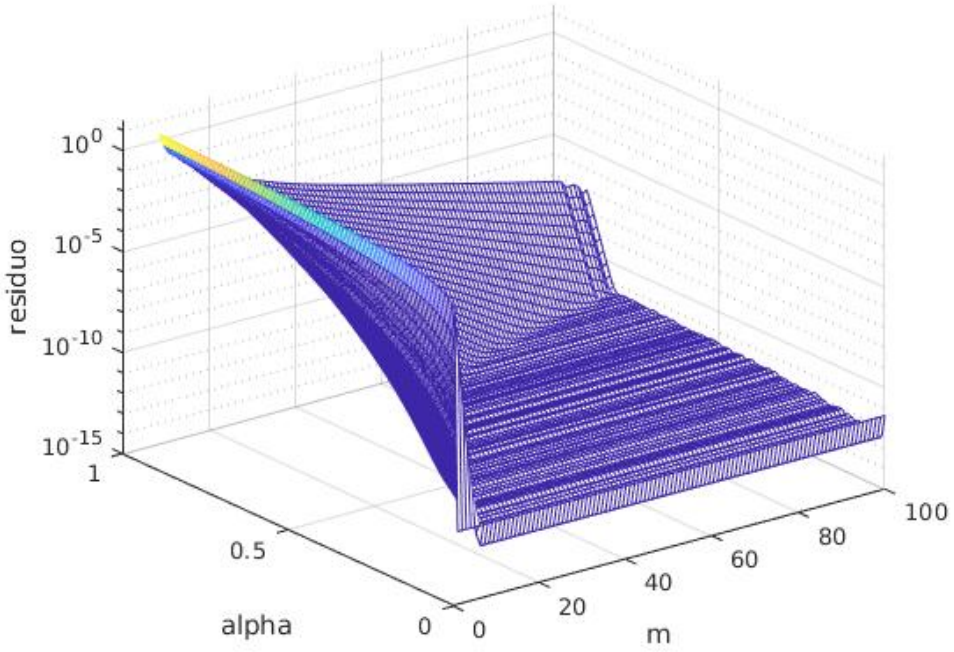


Figura 3.2: Residuo dell'approssimazione $\mathbf{x}(\alpha)$ ottenuta con il modello di ordine ridotto al variare di α e m

Il grafico appena visto è stato ottenuto con i dati della matrice H_{SB} di dimensioni 683446 x 683446 (si veda il prossimo capitolo). Come si può osservare dalla figura, il residuo diminuisce al diminuire di α e al crescere di m, come ci si sarebbe aspettato. Notiamo che in questo caso una scelta di $m > 50$ dà un residuo basso anche per valori consistenti di α ($0.5 \leq \alpha \leq 1$).

3.3 Restarted Arnoldi

Un'ultima cosa da analizzare prima di procedere con gli esperimenti numerici è l'occupazione di memoria. Così come il modello di ordine ridotto, la maggior parte dei metodi basati sull'algoritmo di Arnoldi e finalizzati alla risoluzione di sistemi lineari del tipo $(A - \sigma I)\mathbf{x} = \mathbf{b}$ per vari valori di σ, cercano la soluzione nello spazio di Krylov $K_m(A, \mathbf{v})$ dove $\mathbf{v}$ è il residuo iniziale del sistema. Per garantire un residuo piccolo questo spazio spesso ha dimensioni molto grandi, occupando una quantità non indifferente di memoria. Pertanto sarebbe opportuno riavviare l'algoritmo prendendo il residuo corrente come residuo iniziale. Osserviamo tuttavia che nel caso di un sistema lineare traslato la proprietà di invarianza per traslazione dello spazio di Krylov è mantenuta dopo il riavvio solo se il nuovo spazio di Krylov, ottenuto rispetto al nuovo residuo, è lo stesso per tutti i sistemi traslati, cioè per ogni σ. Questo accade quando i vettori generatori dello spazio di Krylov, che nel nostro caso sono i residui correnti, sono colineari. Per questo motivo introduciamo il Restarted Arnoldi [10].

Vediamo come funziona questo algoritmo applicato al nostro caso.
Fissato un valore di m molto basso come $m = 5$ o 10 e un vettore iniziale $\mathbf{x}_0 = 0$, si definisce il residuo iniziale come $\mathbf{r}_0 = \mathbf{v} - (I - \alpha H^T)\mathbf{x}_0 = \mathbf{v}$ e si considera lo spazio di Krylov $K_m(H^T, \mathbf{v})$. Dopo aver applicato il modello di ordine ridotto, se la soluzione ottenuta non è soddisfacente si ripete il procedimento usando come nuovo residuo iniziale il residuo corrente $\mathbf{r} = \mathbf{v} - (I - \alpha H^T)\mathbf{x}$, dove $\mathbf{x}$ è l'approssimazione della soluzione appena trovata. La potenza di questo metodo risiede nel fatto che il residuo $\mathbf{r}$ è un multiplo del vettore $\mathbf{w}_{m+1}$ come avevamo già dimostrato nel Teorema 12, dunque è evidente che per ogni "restart" i residui relativi ai vari α e il vettore $\mathbf{w}_{m+1}$ sono colineari. Il Restarted Arnoldi risulta dunque particolarmente adatto per risolvere il nostro problema, in quanto, grazie alla proprietà di avere i residui colineari $\forall\alpha$, siamo in grado di riavviare il modello di ordine ridotto trovando una buona approssimazione di $\langle\boldsymbol{\pi}\rangle$ con un'occupazione di memoria decisamente inferiore rispetto al caso precedente. Si veda la sezione "Algoritmi" per il codice Matlab di questo metodo.

Capitolo 4

Esperimenti numerici

Nei capitoli precedenti abbiamo studiato vari algoritmi finalizzati al calcolo del vettore di Page Rank. Ora mettiamoli a confronto per determinare chi sia il più efficiente, confrontando sia il numero di iterazioni necessarie alla convergenza sia il tempo impiegato e la memoria utilizzata.

4.1 I dati

Introduciamo innanzitutto i dati che andremo ad utilizzare. Tutte le matrici che studieremo sono disponibili sul sito https://sparse.tamu.edu/.

Nome	n. pagine	n.link	Descrizione
H_{har}	500	2636	Harvard Web
H_{wiki}	8 297	103 689	Wiki vote
H_{stan}	281 903	2 312 497	Stanford Web
H_{SB}	683 446	7 583 376	Stanford-Berkley web
H_{google}	916 428	5 105 039	Google web

Tutti i risultati presentati in questo capitolo sono stati ottenuti ponendo la tolleranza uguale a 1e-8, il numero massimo di iterazioni uguale a 300 e $\mathbf{v}=\mathbf{e}/\mathrm{n}$. Come criterio di convergenza è stato scelto

$$\frac{\|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\|_1}{\|\mathbf{x}\|_1} < tol,$$

per il metodo delle Potenze, mentre per il metodo di Jacobi

$$\frac{\|\alpha H^T \mathbf{x}^{(i)} - \mathbf{v}\|_1}{\|\mathbf{x}^{(i)}\|_1} < tol.$$

Soffermiamoci un momento sulla matrice H_{har} che rappresenta le pagine con dominio harvard.edu : essendo di dimensioni notevolmente ridotte rispetto alle altre è conveniente utilizzarla per verificare in maniera rapida alcuni concetti introdotti nei capitoli

precedenti.
Innanzitutto ricordiamo che la figura 1.2 ne mostra lo spettro. Inoltre può essere interessante vederne il "pattern" per avere una prima idea della sparsità di questa matrice.

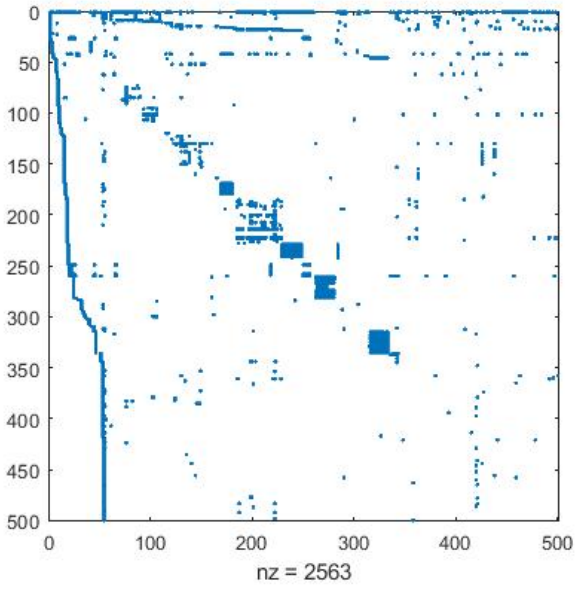


Figura 4.1: Spy della matrice dei link del web di Harvard

Infine applicando gli algoritmi visti otteniamo che le prime 10 pagine più importanti sono:

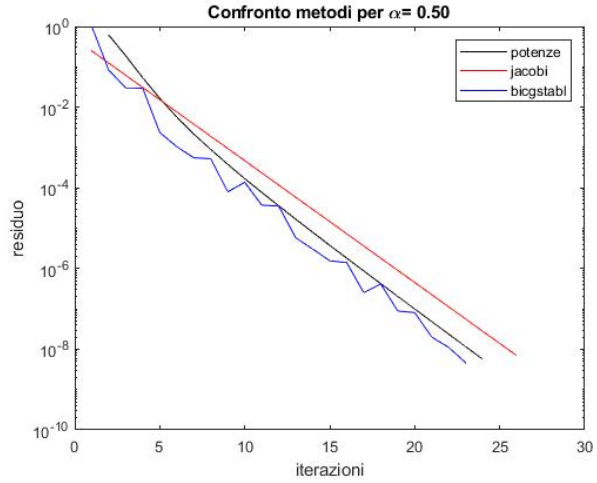
indice	pagerank	url
1	0.0843	http://www.harvard.edu
10	0.0167	http://www.hbs.edu
42	0.0166	http://search.harvard.edu:8765/custom/query.html
130	0.0163	http://www.med.harvard.edu
18	0.0139	http://www.gse.harvard.edu

Come prevedibile la pagina più importante è la homepage del sito web di Harvard e le successive sono le homepage dei singoli corsi di studio presenti nell'università come economia e medicina (si veda [11]).

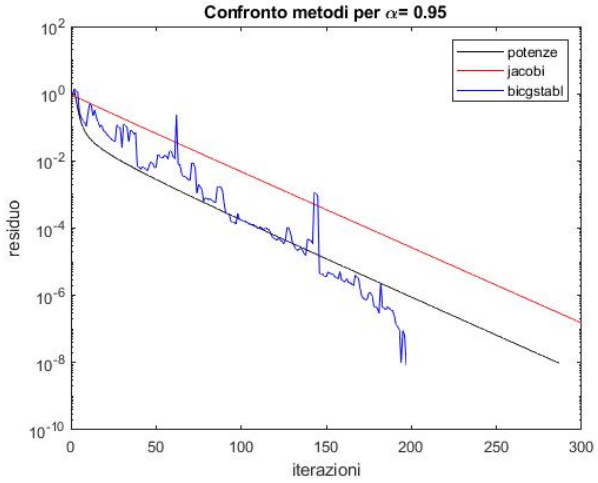
4.2 Convergenza dei metodi iterativi

In questa sezione confronteremo i tempi di convergenza e il numero di iterazioni dei tre algoritmi "classici" che abbiamo visto: il metodo delle Potenze, il metodo di Jacobi e il metodo del Bi-Gradiente Coniugato Stabilizzato attraverso la funzione Matlab `bicgstabl`.

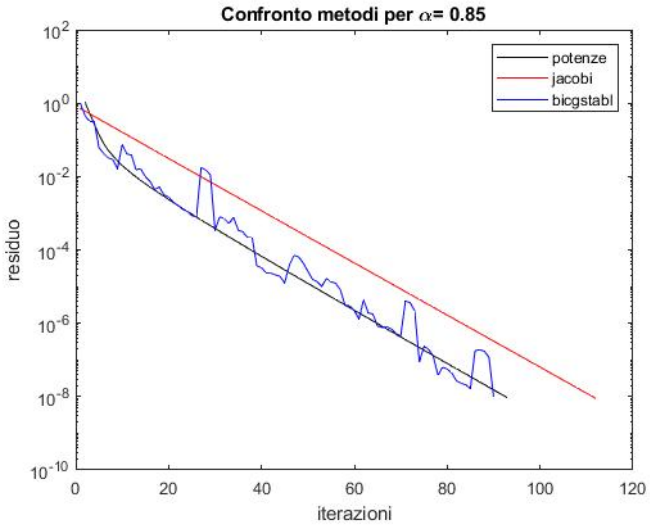
Innanzitutto confrontiamo graficamente il numero di iterazioni per la convergenza di questi tre metodi applicati alla matrice H_{SB}.



(a) α=0.5



(b) α=0.95



(c) α=0.85

Figura 4.2: Confronto della convergenza dei diversi metodi per vari α sulla matrice H_{SB}

Possiamo osservare che il metodo di Jacobi necessita del maggior numero di iterazioni per giungere a convergenza. Il metodo delle Potenze e `bicgstabl` sono confrontabili nei casi $\alpha = 0.5$ e $\alpha = 0.85$, mentre nel caso $\alpha = 0.95$ il metodo `bicgstabl` è notevolmente più efficiente del metodo delle Potenze. Notiamo inoltre che Jacobi e Potenze hanno lo stesso andamento di α^k. Questi grafici sono stati ottenuti applicando i metodi visti alla matrice H_{SB}, tuttavia si ottengono risultati analoghi anche con le matrici di dati H_{har},H_{google} e H_{stan}. L'unica differenza si può notare con H_{google}, matrice di dimensioni maggiori di H_{SB}, dove anche nei casi $\alpha = 0.85$ e $\alpha = 0.5$ la distanza tra `bicgstabl` e potenze è notevole. Questo sarà evidente osservando il numero di iterazioni raccolte

nelle prossime tabelle.

Guardiamo ora anche i tempi di convergenza:

α	Potenze	Jacobi	`bicgstabl`
0.5	0.5096 s (24 it)	0.5266 (26 it)	0.8375 s (23 it)
0.85	1.6996 s (93 it)	1.8111 s (112 it)	2.8893 s (90 it)
0.95	5.0001 s (287 it)	4.7285 s (300 it)	6.2103 s (197 it)

Tabella 4.1: Convergenza dei metodi applicati alla matrice H_{SB}

Come già osservato il metodo `bicgstabl` necessita di un numero di iterazioni decisamente minore, in particolare questo è evidente per $\alpha = 0.95$. Tuttavia, analizzando anche i tempi di convergenza, quest'ultimo metodo risulta essere il più lento. Il metodo delle Potenze e Jacobi hanno tempi confrontabili, ma in particolare Jacobi è lievemente più veloce per $\alpha = 0.95$.

α	Potenze	Jacobi	`bicgstabl`
0.5	1.1234 s (23 it)	1.1462 s (24 it)	1.4881 s (19 it)
0.85	3.8620 s (90 it)	5.0676 s (101 it)	4.8689 s (61 it)
0.95	11.2809 s (279 it)	11.1059 s (300 it)	8.4198 s (131 it)

Tabella 4.2: Convergenza dei metodi applicati alla matrice H_{google}

Ricordiamo che la matrice H_{google} ha dimensioni 916428×916428, mentre H_{SB} è 683446×683446, pertanto vediamo come variano i dati applicando i metodi ad una matrice di dimensione maggiore. Notiamo che il numero di iterazioni è rimasto pressoché invariato; lo stesso non succede per i tempi di convergenza i quali, invece, sono quasi raddoppiati. Anche in questo caso il metodo `bicgstabl` risulta essere molto conveniente per valori alti di α.

α	Potenze	Jacobi	`bicgstabl`
0.5	0.3418 s (22 it)	0.3733s (26 it)	0.5488 s (29 it)
0.85	1.0736 s (85 it)	1.3307 s (109 it)	1.7523 s (102 it)
0.95	3.2292 s (272 it)	3.4329 s (300 it)	4.6859 s (277 it)

Tabella 4.3: Convergenza dei metodi applicati alla matrice H_{stan}

In questo caso si nota che il metodo `bicgstabl` è il più lento, ma, contrariamente ai casi precedenti, il metodo delle Potenze necessita di un numero di iterazioni minore rispetto agli altri due metodi.

4.3 Convergenza dei metodi per calcolare il vettore di PageRank atteso

L'obiettivo di questa sezione è confrontare vari metodi finalizzati tutti al calcolo del vettore di PageRank atteso $\langle \boldsymbol{\pi} \rangle$. Per quanto visto nella sezione 2.2.1, fissata una "griglia di α" è possibile approssimare il vettore di PageRank atteso con

$$\langle \boldsymbol{\pi} \rangle \approx \sum_{i=1}^{k} \pi(\alpha_i) f(\alpha_i) \left[x_i - x_{i-1} \right]. \tag{4.1}$$

Identifichiamo $\alpha_i = i \; \forall i = 1, 2, \ldots, k$. Sia ora g la funzione di densità della distribuzione di Poisson:

$$g(i) = \frac{\lambda^i}{i!} e^{-\lambda} \quad \forall i = 1, 2, \ldots, k$$

Richiediamo che questa funzione di densità dia più importanza a valori "alti" di α, cioè attorno ad $\alpha = 0.85$, e meno ai valori estremi dell'intervallo $[0, 1]$. Per questo motivo scegliamo

$$f(\alpha_i) = f(i) := g(k + 1 - i), \quad \lambda = 1 - 0.85 = 0.15$$

Dunque la somma di Riemann (4.1) può essere riscritta come

$$\langle \boldsymbol{\pi} \rangle \approx \sum_{i=1}^{k} \pi(\alpha_i) \frac{\lambda^{k+1-i}}{(k+1-i)!} e^{-\lambda} \left[x_i - x_{i-1} \right]$$

Ognuno dei metodi visti è in grado di calcolare $\boldsymbol{\pi}(\alpha)$ per ogni valore di α e di conseguenza anche $\langle \boldsymbol{\pi} \rangle$. In particolare confronteremo il Restarted Arnoldi, il modello di ordine ridotto, il metodo delle Potenze Ottimizzato, il metodo delle Potenze classico e il metodo di Jacobi applicati alle matrici H_{har}, H_{wiki}, H_{SB} e H_{google} rispetto a varie griglie di valori di α.

Matrice	Restarted Arnoldi	Ordine Ridotto	Potenze Ottimizzato	Potenze	Jacobi
H_{har}	0.0175 s (m = 10, rest = 3)	0.0186 s (m = 40)	0.0288 s (2120 it)	0.0217 s (2421 it)	0.4305 s (3317 it)
H_{wiki}	0.0281 s (m = 10, rest = 2)	0.0263 s (m = 20)	0.3684 s (1011 it)	0.3691 s (1269 it)	0.8771 s (1597 it)
H_{SB}	2.6616 s (m = 5, rest = 6)	11.8757 s (m = 90)	63.3259 s (2508 it)	71.3863 s (2901 it)	72.7807 s (3291 it)
H_{google}	3.8814 s (m = 5, rest = 6)	14.2314 s (m = 75)	113.7179 s (2554 it)	122.858 s (2824 it)	114.4353 s (3033 it)

Tabella 4.4: Tempi di convergenza per l'approssimazione di $\langle \boldsymbol{\pi} \rangle$ per $\alpha \in \{0.00, 0.01, \ldots, 0.89, 0.90\}$

E' notevole la velocità computazionale del modello di ordine ridotto e del Restarted Arnoldi rispetto agli altri metodi. Questo perché

$$\langle \mathbf{x} \rangle \approx \sum_{i=1}^{91} x(\alpha_i) \frac{\lambda^{92-i}}{(92-i)!} e^{-\lambda} \left[x_i - x_{i-1} \right] = 0.01 e^{-\lambda} \lambda^{92} \sum_{i=1}^{91} x(\alpha_i) \frac{\lambda^{-i}}{(92-i)!} =$$

$$= 0.01 e^{-\lambda} \lambda^{92} \sum_{i=1}^{91} W \mathbf{a}(\alpha) \frac{\lambda^{-i}}{(92-i)!} = 0.01 e^{-\lambda} \lambda^{92} W \sum_{i=1}^{91} \mathbf{a}(\alpha) \frac{\lambda^{-i}}{(92-i)!}$$

dove $\lambda = 0.15$ e $\mathbf{a}(\alpha) = (I - \alpha U)^{-1} \mathbf{e}_1$. Pertanto a livello computazionale la parte più consistente è il calcolo del vettore $\mathbf{a}(\alpha)$ $\forall \alpha$ e della matrice W, il quale viene eseguito un'unica volta per ogni valore di α.

Matrice	Restarted Arnoldi	Ordine Ridotto	Potenze Ottimizzato	Potenze	Jacobi
H_{har}	0.0298 s (m = 5, rest = 5)	0.0162 s (m = 40)	0.081 s (7964 it)	0.1871 s (23771 it)	2.2176 s (32357 it)
H_{wiki}	0.0832 s (m = 10, rest = 2)	0.0238 s (m = 20)	1.6386 s (3623 it)	4.1179 s (12559 it)	7.5138 s (15799 it)
H_{SB}	19.2442 s (m = 10, rest = 4)	20.0536 s (m = 110)	567.7166 s (21122 it)	704.2111 s (28360 it)	714.9788 s (32126 it)
H_{google}	27.4355 s (m = 10, rest = 4)	15.9369 s (m = 80)	918.0182 s (19658 it)	1090.5535 s (27611 it)	966.0914 s (29623 it)

Tabella 4.5: Tempi di convergenza per l'approssimazione di $\langle \boldsymbol{\pi} \rangle$ per $\alpha \in \{0.00, 0.001, \ldots, 0.899, 0.90\}$

In questo tabella abbiamo raccolto dati per molti più valori di α rispetto al primo caso, pertanto avremo una approssimazione più accurata del valore di PageRank atteso $\langle \boldsymbol{\pi} \rangle$. L'accuratezza della soluzione va a discapito però della velocità e dell'occupazione di memoria, vediamo infatti che i tempi di convergenza di tutti i metodi hanno subito un leggero incremento così come la dimensione dello spazio di Krylov relativo alle matrici H_{SB} e H_{google}.

Matrice	Restarted Arnoldi	Ordine Ridotto	Potenze Ottimizzato	Potenze	Jacobi
H_{har}	0.0049 s (m = 10, rest = 3)	0.0065 s (m = 50)	0.0390 s (3314 it)	0.0342 s (4151 it)	0.4593 s (5782 it)
H_{wiki}	0.0234 s (m = 5, rest = 4)	0.0161 s (m = 20)	0.4307 s (1116 it)	0.5102 s (1521 it)	0.8989 s (2011 it)
H_{SB}	2.5981 s (m = 5, rest = 7)	78.4314 s (m= 230)	105.101 s (4525 it)	135.6116 s (5166 it)	117.5252 s (5745 it)
H_{google}	3.6097 s (m = 5, rest = 7)	84.3578 s (m = 200)	164.9532 s (4495 it)	181.5734 s (5059 it)	157.171 s (5383 it)

Tabella 4.6: Tempi di convergenza per l'approssimazione di $\langle \boldsymbol{\pi} \rangle$ per $\alpha \in \{0.00, 0.01, \ldots, 0.98, 0.99\}$

Matrice	Restarted Arnoldi	Ordine Ridotto	Potenze Ottimizzato	Potenze	Jacobi
H_{har}	0.0350 s (m = 5, rest = 6)	0.0195 s (m = 60)	0.0734 s (8144 it)	0.2809 s (42826 it)	2.4462 s (56433 it)
H_{wiki}	0.0890 s (m = 10, rest = 2)	0.0207 s (m = 20)	1.649 s (3803 it)	4.9676 s (15063 it)	8.4488 s (19897 it)
H_{SB}	22.0622 s (m = 10, rest = 4)	104.3327 (m=300)	845.7034 s (37210 it)	1064.6679 s (53015 it)	1086.543 s (58787 it)
H_{google}	39.638 s (m = 10, rest = 4)	201.5767s (m=300)	>15 min	>15 min	>15 min

Tabella 4.7: Tempi di convergenza per l'approssimazione di $\langle \boldsymbol{\pi} \rangle$ per $\alpha \in \{0.00, 0.001, \ldots, 0.989, 0.99\}$

Oltre ai tempi di convergenza estremamente alti dovuti alla quantità di α studiati, notiamo che il Restarted Arnoldi impiega leggermente più tempo del modello di ordine ridotto per matrici di dimensione piccole, mentre per matrici più grandi è estremamente vantaggioso non solo dal punto di vista di memoria occupata ma anche per la velocità di convergenza, arrivando ad essere fino a 28 volte più veloce del modello di ordine ridotto come si può vedere nella Tabella 4.6. I metodi classici come il metodo delle Potenze e Jacobi risultano poco appropriati per la gestione di problemi di queste grandi dimensioni, restituendo tempi di convergenza molto elevati.

Conclusioni

In questa tesi sono stati analizzati vari metodi finalizzati al calcolo del vettore di PageRank. Il problema è stato introdotto enunciando le prime nozioni teoriche necessarie per poi compiere una serie di modifiche del modello di partenza in modo da garantire la convergenza e la buona posizione del problema.
Successivamente si è notata una poliedricità di questo problema: può essere visto sia come un problema agli autovalori $G^T\boldsymbol{\pi} = \boldsymbol{\pi}$ che come risoluzione del sistema lineare $\left(I - \alpha H^T\right)\boldsymbol{\pi} = \mathbf{v}$. Soffermandosi su questa dualità la tesi si sviluppa nel confronto tra questi due diversi approcci.
Nel capitolo 2 sono stati seguiti i primi passi dei fondatori di Google, Brin e Page, applicando il metodo delle Potenze, metodo alla base di ogni algoritmo finalizzato alla ricerca di un'autocoppia. Assumendo poi che il parametro di teletrasporto α sia una variabile stocastica, si è considerato il vettore di PageRank $\boldsymbol{\pi}$ come una funzione dipendente da α e cercando di approssimare il vettore di Page Rank atteso $\langle\boldsymbol{\pi}\rangle = \int_0^1 \boldsymbol{\pi}(\alpha)f(\alpha)d\alpha$, è stata proposta un'ottimizzazione del metodo delle potenze. Questo nuovo algoritmo, sfruttando la proprietà del metodo delle potenze che garantisce una convergenza più rapida nel caso in cui il vettore iniziale sia vicino alla soluzione cercata, va a scegliere come vettore iniziale il vettore di PageRank trovato attraverso il metodo delle potenze classico e corrispondente ad un diverso valore di α.
Dimostrando con dei passaggi algebrici come passare dal problema agli autovalori iniziale ad un sistema lineare, nel capitolo 4 si è studiata l'applicazione del metodo classico di Jacobi alla matrice sparsa H. Il focus di questo elaborato risiede nell'ultima sezione del capitolo 4 dove è stato introdotto il modello di ordine ridotto. Questo metodo basato sull'algoritmo di Arnoldi è in grado di calcolare $\boldsymbol{\pi}(\alpha)$ per vari valori di α in un modo veramente efficiente. Le proprietà che rendono questo metodo così efficace sono tre: la proprietà di invarianza per traslazione degli spazi di Krylov, la garanzia che il vettore $\boldsymbol{\pi}$ appartenga allo spazio di Krylov $K_m(H^T, \mathbf{v})$ e la peculiarità dell'algoritmo di Arnoldi di fornire una base ortonormale di questo spazio. Grazie a questi tre elementi è stato possibile dimostrare che il residuo del sistema lineare $(I - \alpha H^T)\mathbf{x} = \mathbf{v}$ appartiene allo spazio di Krylov $K_{m+1}(H^T, \mathbf{v})$: pertanto scegliendo in modo opportuno un vettore $\mathbf{u}$ che approssimi $(I - \alpha H^T)\mathbf{u} \approx \mathbf{v}$ è possibile ottenere un residuo molto basso. In particolare la scelta ottimale è $\mathbf{u} = W\mathbf{a}(\alpha)$ dove W è la matrice con colonne ortonormali generata da Arnoldi e $\mathbf{a}(\alpha) = \|\mathbf{v}\|_2 \left(I - \alpha U\right)^{-1}\mathbf{e}_1$. Il grande vantaggio di questo metodo risiede

nel fatto che Arnoldi vada applicato un'unica volta e che per ogni valore di α sia richiesto solo il calcolo di $\mathbf{a}$ che consiste nel calcolo dell'inversa della matrice $(I - \alpha U)^{-1}$ di dimensioni $m \times m$, dove $m \ll n$.
Come si può appurare dai dati raccolti nella sezione "Esperimenti Numerici" quest'ultimo metodo arriva ad essere fino a 60 volte più veloce dei metodi classici, ma occupa molta memoria e, considerate le dimensioni delle matrici studiate, questo può rappresentare un problema consistente. Per far fronte al problema di memoria è stata dunque applicata una modifica "restartata" di questo metodo, ovvero il Restarted Arnoldi. I dati raccolti confermano che questo metodo è in grado di convergere alla soluzione molto velocemente richiedendo una dimensione estremamente ridotta dello spazio di Krylov, attorno a $m = 5$ o 10.

Bibliografia

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

[2] A.N.Langville and Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engines Rankings*. Princeton University Press, William Street,Princeton, New Jersey, 2006.

[3] Carl D.Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, Philadelphia, 2000.

[4] V.Simoncini e D.Palitta. Dispense del corso di calcolo numerico, modulo di algebra lineare numerica. 2016.

[5] V.Simoncini. Dispense del corso di matematica computazionale. 2021.

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

[7] G.H.Golub and C.Greif. *An Arnoldi-type algorithm for computing PageRank*. University of British Columbia, Vancouver, B.C. V6T 1Z4, Canada, 2006.

[8] M.Bianchini, M.Gori, and F.Scarselli. *Inside PageRank. ACM Transactions on InternetTechnology*. University of Siena, 2005.

[9] H.De Looij. Google pagerank and reduced-order modelling. Master's thesis, Delft University of Technology, 2013.

[10] V.Simoncini. *Restarted Full Orthogonalization Method for shifted linear systems*. BIT Numerical Mathematics, v.43, n.2, pp. 459-466, 2003.

[11] C.Moler. Matlab - google page rank, capitolo 7. 2011.

Algoritmi

```
    function [x_exp,pi_exp]=ReducedOrder(H,alphas,m,v)
n=length(H);
e1=[1;zeros(m-1,1)];
s=zeros(m,1);
%pi=zeros(n,length(alphas)); %pi è una matrice con sulle colonne i valori valori di pi(alpha)

%innanzitutto costruiamo una base ortonormale usando Arnoldi
[W,U]=Arnoldi(H',m,v);

%ora calcoliamo l'approssimazione di x(alphas) nello spazio di Krylov per ogni valore di alpha.
for i=1:length(alphas)
    a=(eye(m)-alphas(i)*U)\e1; %a(alpha)
    s=s+a;
    %calcoliamo il residuo
    %r=alpha*U(m+1,m)*A(m);  chi è U(m+1,m)?
end
x_exp=W*s;
pi_exp=x_exp/sum(x_exp);
end
```

```
function [x_media,errore]=fom_shift(a,b,m,maxit,tol,shift)
% restarted fom for shifted systems:
%       (I - shift(j)*a) x = b
%
% a     nxn matrix
% b     rhs = right-hand side of the equation above.
% m     max krylov sub. dimension (e.g., 20)
% maxit max number of restarts (e.g., 100)
% tol   convergence tolerance (e.g., 1e-8)
% shift shift values (real or complex)

ns=length(shift);
[n,n]=size(a);
m1=m+1;
tet=10.;
itera=0; x=zeros(n,ns);
res=b; arr=norm(res)*ones(1,ns); init_res=arr(1);
erro=1; errore=[];
V=res/init_res;
inoconv=1:ns;

while (max(erro) > tol  & itera < maxit);

   itera=itera+1;
   H=zeros(m+1,m,ns);
   [mar,imax]=min(arr(inoconv)); %minimo dei residui per vari alpha
```

```
   V=V(:,end);
   i=0;

   while (i<m & max(erro)>tol )

     i=i+1;
     i1=i+1; it = 0; t=0.;hinorm=0.;

     V(:,i1) = a*V(:,i); %creo la matrice V con m+1 colonne

% Double Gram-Schmidt process
     while ( t*tet <= hinorm & it < 2 )
       hinorm=0.; it = it+1;
       for j=1:i
            t = V(1:n,j)'*V(1:n,i1); %prodotto scalare tra colonne di v con colonna i1 di V
            hinorm = hinorm + abs(t^2); H(j,i,:)=H(j,i,:)+t;
            V(1:n,i1)=V(1:n,i1)-t*V(1:n,j); %tolgo la proiezione
       end
       t = norm(V(1:n,i1));
     end
     H(i1,i,:)=t; if (t ~= 0.) V(1:n,i1)=V(1:n,i1)/t; end %normalizzo la soluzione trovata
    %for k=1:ns,H(i,i,k)=H(i,i,k)+shift(k);end
   end

   erro=0;
   y=zeros(i,ns);
% Solve projected problem
  for k=1:length(inoconv)
       y(1:i,inoconv(k))=(eye(m)-shift(k)*H(1:m,1:m,inoconv(k)))\(arr(inoconv(k))*eye(m,1)); %(I-aH)/||arr(k)||;
       erro=max(erro,abs(shift(k)*H(i1,i)*y(i,inoconv(k)))/init_res);
  end
% True residual
% res = b*ones(1,ns) - a*x-x*spdiag(shift);
  x_media=sum(x(:,inoconv),2)+V(:,1:i)*sum(y(1:i,inoconv),2);
  arr=abs(-(H(i1,i)*y(i,:).*shift(:)'));
  erro=(arr)/init_res;

% Eliminate converged systems
  [val,inoconv]=find(erro>tol);
  errore=[errore;erro];

end;

fprintf('\n')
fprintf('fom: rest. %g  max res %g  m %g num.shifts %g \n',itera,max(arr),m,ns);
%semilogy(errore);
%xlabel('number of restarts')
%ylabel('relative residuals')
```