Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

Metodi randomizzati per la risoluzione del problema di Pagerank

Tesi di Laurea in Analisi Numerica

Relatore:
Chiar.mo Prof.
Davide Palitta

Presentata da:
Davide Battani

Anno Accademico 2023-2024

Indice

Introduzione **3**

Notazioni, definizioni e risultati preliminari **5**

1 Il problema di PageRank **7**
1.1 Introduzione al problema di PageRank 7
1.2 Il metodo delle potenze 11
1.3 Filter-Based Adaptive PageRank 12
1.4 Modified Filter-Based Adaptive PageRank 14

2 Metodi randomizzati **16**
2.1 Approssimazione di Range(A) 16
2.2 Metodi di proiezione 18
2.3 Metodi di proiezione randomizzati 20

3 Esperimenti numerici **22**
3.1 Esperimenti su Adaptive Pagerank 22
3.2 Esperimenti sui metodi randomizzati 27

Conclusioni **30**

Introduzione

Pagerank è l'algoritmo di analisi ideato da Larry Page e Sergey Brin, i fondatori di Google, con lo scopo di valutare l'importanza di un'enorme quantità di pagine web in maniera veloce, frequente, e soprattutto senza fare ricorso all'intervento umano. In questa tesi parlerò di alcuni metodi utilizzabili per la risoluzione del problema di Pagerank, focalizzandomi in particolare su una nuova tipologia di metodi randomizzati. Nel primo capitolo introdurrò il problema di Pagerank, parlando dello scopo dell'algoritmo, del metodo di risoluzione classico, e di alcuni metodi adattivi introdotti successivamente per aumentare l'efficienza dell'algoritmo. Nel secondo capitolo parlerò invece dei metodi randomizzati, una nuova categoria di metodi che si basano sulla costruzione casuale di uno spazio di piccole dimensioni, in cui verrà ricercata un'approssimazione della soluzione del problema iniziale. Nell'ultimo capitolo ho svolto infine esperimenti numerici su diversi dataset, e ho confrontato tra loro i risultati ottenuti.

Notazioni, definizioni e risultati preliminari

Enunciamo ora alcune definizioni e alcuni teoremi che utilizzeremo all'interno di questa tesi.

Definizione 0.1. Una *matrice di permutazione* è una matrice ottenuta scambiando alcune righe e colonne di una matrice identità.

Definizione 0.2. Una matrice $A \in \mathbb{R}^{n\times n}$ si dice *riducibile* se esistono una matrice di permutazione T, e due matrici quadrate X e Z tali che

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

Una matrice non riducibile è detta *irriducibile*.

Definizione 0.3. Un *grafo diretto* G consiste in una coppia (V, E) formata da un insieme di *vertici* V e un insieme di *archi* $E \subseteq V \times V$. In particolare, un grafo diretto è *pesato* se ad ogni suo arco è associato un valore numerico detto *peso*.

Definizione 0.4. Dato un grafo diretto $G = (V, E)$, con V insieme finito di vertici, la sua *matrice di adiacenza* è la matrice

$$A = (a_{i,j}) = \begin{cases} 1 & \text{se } (v_i, v_j) \in E, \\ 0 & \text{se } (v_i, v_j) \notin E. \end{cases}$$

Definizione 0.5. La matrice *pesata* di un grafo diretto pesato $G = (V, E)$ è una variante della matrice di adiacenza in cui ad ogni entrata $a_{i,j}$ della matrice viene assegnato il valore numerico dell'arco (v_i, v_j), se esso appartiene ad E, o il valore 0 altrimenti.

Definizione 0.6. Dati due vertici v e w di un grafo diretto $G = (V, E)$, un *cammino* nel grafo da v a w è una sequenza finita di vertici $v_0, \ldots, v_n$ tali che $v_0 = v$, $v_n = w$, e per ogni i appartenente all'insieme $\{0, \ldots, n-1\}$, (v_i, v_{i+1}) è un arco del grafo.

Il grafo diretto di una matrice A si dice *fortemente connesso* se per ogni coppia di vertici (v, w) del grafo, esiste un cammino nel grafo da v a w.

Teorema 0.1. *Sia $A \in \mathbb{R}^{n\times n}$. La matrice A è irriducibile se e solo se il suo grafo diretto associato è fortemente connesso.*

Definizione 0.7. Una matrice $H \in \mathbb{R}^{n\times n}$ si dice *stocastica per colonne* quando tutti i suoi termini sono non negativi, e la somma dei termini di ciascuna colonna è uguale ad 1.

Teorema 0.2 (variante del teorema di Perron-Frobenius)**.** *Sia $A \in \mathbb{R}^{n\times n}$ una matrice irriducibile e stocastica per colonne, e siano $\{\alpha_1, \ldots, \alpha_n\}$ i suoi autovalori. Allora il suo autovalore dominante in modulo è $\alpha_1 = 1$, l'unico autovettore associato ad esso ha componenti tutte positive, ed è l'unico autovettore della matrice che ha tutte le componenti non negative.*
Inoltre, se tutti i termini di A sono positivi, allora

$$|\alpha_i| < 1, \quad \textit{per ogni } i = 2, \ldots, n.$$

Definizione 0.8. Sia $A \in \mathbb{R}^{n\times m}$. *Range*(A), anche noto come *immagine* di A, è il sottospazio vettoriale di $\mathbb{R}^n$ generato dalle colonne di A. *Ker*(A), anche noto come *[nucleo]* di A, è l'insieme dei vettori $v \in \mathbb{R}^n$ tali che $Av = 0$.

Definizione 0.9. Un *proiettore* P è una matrice $P \in \mathbb{R}^{n\times n}$ idempotente, ossia tale che $P^2 = P$.

Enunciamo alcune proprietà dei proiettori:

Proposizione 0.1. *Sia P un proiettore. Allora anche $I - P$ è un proiettore, e Ker(P) = Range$(I - P)$.*

Proposizione 0.2. *Ker$(P) \cap$ Range$(P) = \{0\}$. Inoltre, $\mathbb{R}^n =$ Ker$(P) \oplus$ Range(P).*

Proposizione 0.3. *Ogni coppia di sottospazi vettoriali C, D che forma una somma diretta di $\mathbb{R}^n$, definisce un unico proiettore P tale che Range$(P) = C$ e Ker$(P) = D$.*

Definizione 0.10. Sia P un proiettore ortogonale, e V una matrice le cui colonne formano una base ortogonale di $C =$ Range(P). La matrice $P = VV^T$ è la *rappresentazione matriciale di un proiettore ortogonale.*

Si osservi che la matrice VV^T è idempotente, ma la sua rappresentazione non è unica, in quanto ogni base ortonormale di Range(P) induce una diversa matrice V con cui costruire tale rappresentazione.

Definizione 0.11. Siano C, D due sottospazi che formano una somma diretta di $\mathbb{R}^n$, con D complemento ortogonale di C. Il proiettore P è *ortogonale su C* se Ker$(P) =$ Range$(P)^{\perp}$.

Capitolo 1

Il problema di PageRank

1.1 Introduzione al problema di PageRank

Quando utilizziamo un motore di ricerca per ottenere informazioni su un determinato argomento, ci viene fornita una lista di numerose pagine contenenti le parole chiave della nostra ricerca; la scelta e l'ordine in cui esse vengono presentate sono il risultato di un processo di ranking di una libreria enorme e in continuo aggiornamento, che avviene però in pochissimo tempo, senza intervento umano, ed è in grado di mettere in evidenza quali siano le pagine più significative e affini alla nostra ricerca.
Specialmente per ricerche molto generiche, l'elenco ottenuto non si basa prevalentemente sulla corrispondenza delle parole chiave, privilegiando invece l'importanza delle pagine stesse.

La definizione di "importanza" di una pagina è una definizione ricorsiva: l'importanza di una pagina è data dal numero di pagine collegate ad essa (tramite un outlink), e dalla loro importanza.

Possiamo costruire una formulazione equivalente del problema descritto in termini di una camminata aleatoria su un grafo diretto G. Visualizziamo l'insieme delle pagine da esaminare come un grafo pesato $G = (V, E)$, i cui vertici $V = \{p_1, \ldots, p_n\}$ corrispondono alle pagine web, e dove ogni arco (p_i, p_j) corrisponde alla presenza di un collegamento (o *outlink*) dalla pagina p_i verso la pagina p_j. Per definire poi il valore numerico associato a ciascun arco, è necessario dare prima la seguente definizione.

Definizione 1.1. Il *grado* di una pagina p, è il numero di outlinks presenti nella pagina p, e si indica con $\deg(p)$.

L'idea che ci permette di descrivere il peso degli archi del grafo G, è che quando una pagina p_i contiene outlink verso una pagina p_j, p_i "trasferisce" a p_j una frazione $\frac{1}{\deg(p_i)}$ della propria importanza. Il peso di un arco (p_i, p_j) è dato dunque dal valore $\frac{1}{\deg(p_i)}$.

Introdotto questo concetto, segue che l'importanza $I(p_k)$ di una pagina p_k è data dalla formula

$$I(p_k) = \frac{1}{\deg(p_1)} \cdot I(p_1) + \cdots + \frac{1}{\deg(p_h)} \cdot I(p_h),$$

con $p_1, \ldots, p_h$ che sono tutte le pagine che puntano alla pagina p_k.

Si osservi che ogni pagina può presentare un outlink verso se stessa, e questo outlink contribuisce ad aumentare il grado della pagina.

Definito dunque questo grafo, consideriamo che in un istante k, un "walker" si trovi nella pagina p_i; nell'istante $k+1$ esso si sposterà casualmente in una delle pagine linkate dalla pagina p_i, avendo una probabilità $\frac{1}{\deg(p_i)}$ di andare in ciascuna di esse.

Definizione 1.2. Il *PageRank* di una pagina p è definito come la probabilità che in un istante $k > K$, con $K >> 0$, il walker si trovi nella pagina p.

Definizione 1.3. La *Transition Probability Matrix* (o *matrice di Hyperlink*) è la matrice

$$P = (P_{i,j}) = \begin{cases} \frac{1}{\deg(p_i)}, & \text{se } p_i \to p_j, \\ 0, & \text{altrimenti.} \end{cases}$$

In altre parole, la matrice di Hyperlink è la matrice di adiacenza del grafo pesato appena descritto.

Sfruttando queste definizioni, e costruendo il vettore

$$\begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} = \begin{pmatrix} I(p_1) \\ I(p_2) \\ \vdots \\ I(p_n) \end{pmatrix},$$

ossia il vettore la cui k-esima entrata corrisponde all'importanza della pagina p_k, segue che

$$\begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} = P^T \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix}.$$

Definizione 1.4. Il vettore $\begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix}$ è il *vettore di PageRank*.

Il vettore di PageRank è il vettore contente il dato dell'importanza di tutte le pagine considerate, nonché autovettore associato all'autovalore 1 della trasposta della matrice di Hyperlink: abbiamo dunque riformulato il problema di ranking, riconducendoci ad un problema agli autovalori.
La matrice così costruita potrebbe però presentare alcuni problemi che comprometterebbero l'esistenza e l'unicità della soluzione cercata: in particolare, P potrebbe avere delle righe nulle, corrispondenti a pagine senza alcun outlink. Potrebbe poi non avere autovalore 1, oppure, nel caso in cui 1 sia autovalore di P, potrebbe non essere semplice. Per evitare questi problemi, eseguiamo alcune modifiche alla matrice di Hyperlink al fine di soddisfare le ipotesi del teorema di Perron-Frobenius, assicurando così l'esistenza e

l'unicità del vettore di PageRank.

La prima modifica che apportiamo alla matrice P consiste nella rimozione delle "dangling pages", ossia delle pagine che non contengono nessun outlink, e dove quindi il walker si bloccherebbe; per risolvere questo problema, è sufficiente aggiungere un set completo di transizioni in uscita, a partire dalle pagine in questione.

Definizione 1.5. Il vettore $v = \begin{pmatrix} \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{pmatrix} \in \mathbb{R}^n$ è il *vettore di personalizzazione.*

Identificando dunque le pagine senza outlink tramite il vettore

$$d = (d_i) = \begin{cases} 1, & \text{se } \deg(p_i) = 0, \\ 0, & \text{altrimenti,} \end{cases}$$

eseguiamo la prima trasformazione di P costruendo $D = d \cdot v^T$, e $P' = P + D$.
Si osservi che il vettore v non deve necessariamente fornire una distribuzione di probabilità uniforme, ma potrebbe anche fornire distribuzioni di probabilità che favoriscano alcune pagine, a discapito di altre.
Grazie a questa prima modifica, P' è ora una matrice stocastica per righe.

Formulando il problema nel modo appena descritto, è molto probabile che la matrice P', e conseguentemente la matrice P, sia riducibile; la seconda modifica che apportiamo alla matrice ha dunque lo scopo di garantire che essa sia irriducibile.
Per garantire ciò, dovremo rendere il suo grafo diretto associato fortemente connesso, e il modo più comune per farlo è tramite l'aggiunta di un nuovo set completo di transizioni in uscita, con basse probabilità di transizione, a tutti i nodi del grafo.
Definendo dunque il vettore $\mathbf{1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n$, e la matrice $E = \mathbf{1} \cdot v^T$, e $c \in (0, 1)$, costruiamo la matrice $P'' = c \cdot P' + (1 - c) \cdot E$.

Il valore assegnato a c influenza la risoluzione del problema: per valori di c molto vicini ad 1, la struttura della matrice P' sarà maggiormente preservata, ma il problema potrebbe non essere risolvibile.

È importante enunciare alcuni teoremi sullo spettro della matrice P'', che ci permetteranno di ottenere un'ulteriore informazione sul modo in cui il valore assegnato a c influenza la risoluzione del problema.

Teorema 1.1. *Siano $H \in \mathbb{R}^{n \times n}$ una matrice stocastica per colonne, spec$(H) = \{1, \lambda_2, \ldots, \lambda_n\}$, $c \in (0, 1)$ e*

$$P = cH + (1 - c)\frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

Allora spec$(P) = \{1, c\lambda_2, \ldots, c\lambda_n\}$.

Questo teorema asserisce che l'autovalore 1 associato alla matrice $(P'')^T$ sia semplice, indipendentemente da quale sia la sua molteplicità come autovalore associato alla matrice P', di cui è certamente autovalore, in quanto essendo P' stocastica per righe, vale $P'\mathbf{1} = \mathbf{1}$.

Dimostrazione. Essendo H stocastica per colonne, si osservi $H^T\mathbf{1} = \mathbf{1}$.

Sia $\tilde{\mathbf{1}} = \frac{1}{\sqrt{n}}\mathbf{1}$. Allora $P^T\tilde{\mathbf{1}} = cH^T\tilde{\mathbf{1}} + (1-c)\frac{1}{n}\mathbf{1}\mathbf{1}^T\tilde{\mathbf{1}} = \alpha\tilde{\mathbf{1}} + (1-\alpha)\tilde{\mathbf{1}} = \tilde{\mathbf{1}}$.

Sia poi $U = [\tilde{\mathbf{1}}, U_0] \in \mathbb{R}^{n\times n}$ unitaria; allora, $\tilde{\mathbf{1}}$ e le colonne di U_0 sono ortogonali, in quanto $U^*U = I_n$, dunque $\tilde{\mathbf{1}}^T U_0 = \mathbf{0} \in \mathbb{R}^{1\times(n-1)}$. Inoltre, U^*HU e H sono simili, e per questo motivo hanno lo stesso spettro. Segue che

$$U^*HU = \begin{bmatrix} \tilde{\mathbf{1}}^T H\tilde{\mathbf{1}} & \tilde{\mathbf{1}}^T HU_0 \\ U_0^T H\tilde{\mathbf{1}} & U_0^T HU_0 \end{bmatrix} = \begin{bmatrix} 1 & \tilde{\mathbf{1}}^T U_0 \\ U_0^T H\tilde{\mathbf{1}} & U_0^T HU_0 \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \\ * & U_0^T HU_0 \end{bmatrix}$$

Dato che $\mathrm{spec}(H) = \mathrm{spec}(U^*HU)$, e che per ipotesi $\mathrm{spec}(H) = \{1, \lambda_2, \ldots, \lambda_n\}$, allora gli autovalori $\{\lambda_2, \ldots, \lambda_n\}$ sono gli autovalori nel blocco $T = U_0^* HU_0$.

Si osserva poi che

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

$$= \begin{bmatrix} \alpha & \mathbf{0} \\ * & \alpha T \end{bmatrix} + (1-\alpha)\begin{bmatrix} \tilde{\mathbf{1}}^T\tilde{\mathbf{1}}\tilde{\mathbf{1}}^T\tilde{\mathbf{1}} & \tilde{\mathbf{1}}^T\tilde{\mathbf{1}}\tilde{\mathbf{1}}^T U_0 \\ U_0^*\tilde{\mathbf{1}}\tilde{\mathbf{1}}^T\tilde{\mathbf{1}} & U_0^*\tilde{\mathbf{1}}\tilde{\mathbf{1}}^T U_0 \end{bmatrix} = \begin{bmatrix} \alpha & \mathbf{0} \\ * & \alpha T \end{bmatrix} + (1-\alpha)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0}^T & \mathbf{0}^T\mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \\ * & \alpha T \end{bmatrix}$$

Segue che

$$\mathrm{spec}(P) = \mathrm{spec}(U^*PU) = \{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$$

□

Teorema 1.2. *Sia $H \in \mathbb{R}^{n\times n}$ una matrice stocastica per colonne; se H ha almeno 2 sottoinsiemi chiusi irriducibili (ossia se il suo grafo associato ha due componenti fortemente connesse e distinte), allora il secondo autovalore più grande in modulo della matrice*

$$P = cH + (1-c)\frac{1}{n}\mathbf{1}\mathbf{1}^T$$

è $\lambda_2 = c$.

La matrice P'' conserva la proprietà di stocasticità per righe della matrice P', ma è anche irriducibile e a termini positivi. Segue dunque che la matrice $A = (P'')^T$ sia a sua volta irriducibile e a termini positivi, ma che sia stocastica per colonne. Essa soddisfa dunque le ipotesi del teorema di Perron-Frobenius, nonché le ipotesi dei teoremi appena enunciati, garantendo così che il suo autovalore dominante sia 1, e che esso sia semplice.

In seguito alla riformulazione di questo problema, lo scopo è ora quello di calcolare l'autovettore dominante della matrice A.

1.2 Il metodo delle potenze

Un metodo iterativo particolarmente adatto per il calcolo dell'autocoppia dominante di una matrice $A \in \mathbb{R}^{n\times n}$ è il metodo delle potenze. Stabilendo un vettore iniziale $x^{(0)}$ di norma unitaria, e ponendo $y = Ax^{(0)}$, il metodo consiste nella seguente iterazione:

Algoritmo 1 Metodo delle Potenze

```
1: function STANDARD_POWER_METHOD(A, x^(0), y, tol)
2:     for i = 0, 1, 2, ... do
3:         λ^(i) = (x^(i))^H y
4:         r^(i) = Ax^(i) − λ^(i) x^(i)
5:         if ||r^(i)|| / |λ^(i)| < tol then
6:             Stop
7:         end if
8:         x^(i+1) = y / ||y||
9:         y = Ax^(i+1)
10:    end for
11: end function
```

Una condizione necessaria affinché il metodo converga è che l'autovalore dominante sia semplice, ma ciò è garantito dal teorema di Perron-Frobenius.
Indicando con (λ_1, x_1) l'autocoppia dominante della matrice A, il metodo delle potenze crea due successioni $\{\lambda^{(k)}\}_{k\in\mathbb{N}}$ e $\{x^{(k)}\}_{k\in\mathbb{N}}$ che, per k che tende a $+\infty$, convergono rispettivamente all'autovalore λ_1 e all'autovettore x_1.

Supponiamo ora che A sia diagonalizzabile; esistono dunque le matrici X, la matrice dei suoi autovettori, e D, la matrice diagonale contenente tutti gli autovalori di A, tali che

$$A = XDX^{-1}.$$

Supponiamo poi di conoscere il modulo di tutti gli autovalori della matrice A; in particolare, senza perdere di generalità, supponiamo che $|\lambda_1| < |\lambda_2| \leq \cdots \leq |\lambda_n|$, e sia $y = X^{-1}x^{(0)}$.
Allora osserviamo che vale

$$\frac{x^{(k)}}{\lambda_1^k} = \frac{A^k x^{(0)}}{\lambda_1^k} = \frac{XD^kX^{-1}x^{(0)}}{\lambda_1^k} = \frac{XD^ky}{\lambda_1^k} = \frac{1}{\lambda_1^k}\sum_{i=1}^{n} x_i(\lambda_i)^k(y)_i = \sum_{i=1}^{n} x_i\left(\frac{\lambda_i}{\lambda_1}\right)^k(y)_i =$$

$$= x_1(y)_1 + x_2\left(\frac{\lambda_2}{\lambda_1}\right)^k(y)_2 + \sum_{i=3}^{n} x_i\left(\frac{\lambda_i}{\lambda_1}\right)^k(y)_i \xrightarrow{k\to+\infty} x_1(y)_1.$$

Questo limite ci permette di osservare che la velocità di convergenza del metodo delle potenze dipende dal valore $\left|\frac{\lambda_2}{\lambda_1}\right|$; per valori di questo rapporto più bassi, dunque nel caso in cui il modulo del secondo autovalore sia maggiormente distaccato dal modulo dell'autovalore dominante, il valore $\left|\frac{\lambda_2}{\lambda_1}\right|^k$ tenderà a 0 più rapidamente, e la convergenza del metodo sarà più veloce.

Questa osservazione ci fornisce un'ulteriore informazione su come il valore di c precedentemente utilizzato per costruire la matrice $A = (P'')^T$ influenzi l'efficienza di questo metodo risolutivo: la velocità di convergenza del metodo delle potenze applicato alla matrice A dipende da

$$\left|\frac{\lambda_2(A)}{\lambda_1(A)}\right|^k = |\lambda_2(A)|^k = c^k.$$

Assegnando dunque un valore minore a c, il metodo convergerà più velocemente, ma dobbiamo anche ricordarci che il problema sarà maggiormente condizionato.

1.3 Filter-Based Adaptive PageRank

Analizzeremo ora alcuni metodi adattivi che hanno lo scopo di rendere più efficiente l'algoritmo appena presentato. Da un'analisi più accurata emerge che i tassi di convergenza dei valori di PageRank delle singole pagine non sono uniformi, dunque non è necessario ricalcolare ad ogni iterazione il PageRank delle pagine che sono già converse. Il PageRank della maggior parte delle pagine tende a convergere rapidamente, e le poche pagine che necessitano di più iterazioni sono generalmente le pagine più importanti.

Definizione 1.6. Sia $x_i = I(p_i)$. Il PageRank x_i della pagina p_i è *converso* se

$$\frac{|x_i^{(k+1)} - x_i^{(k)}|}{|x_i^{(k)}|} < \delta. \tag{1.1}$$

Il valore assegnato a δ è arbitrario, e influenza la velocità di calcolo della soluzione. Un valore generalmente appropriato per tale variabile è $\delta \sim 10^{-3}$. [1]

Al fine di velocizzare la computazione del metodo delle potenze, eviteremo di ricalcolare il PageRank delle pagine già converse.
Supponiamo di aver eseguito la k-sima iterazione del metodo, avendo dunque calcolato $x^{(k)}$; dobbiamo ora calcolare $x^{(k+1)}$.
Suddividiamo il set di pagine che stiamo esaminando in due subset C e N, con C che contiene tutte le pagine già converse, e N che contiene quelle delle pagine non ancora converse. Sia $\#N = m$.
Dividiamo A in due sottomatrici: $A_N \in \mathbb{R}^{m\times n}$ e $A_C \in \mathbb{R}^{(n-m)\times n}$. A_N corrisponde alla sottomatrice contenente i collegamenti verso le pagine non converse, A_C contiene quelli verso quelle converse.
Analogamente dividiamo il vettore $x^{(k)}$ in $x_N^{(k)} \in \mathbb{R}^m$ e $x_C^{(k)} \in \mathbb{R}^{(n-m)}$.

Riordinando in blocchi la matrice A e il vettore $x^{(k)}$, possiamo riscriverli nella forma

$$A = \begin{bmatrix} A_N \\ A_C \end{bmatrix}, \quad x^{(k)} = \begin{bmatrix} x_N^{(k)} \\ x_C^{(k)} \end{bmatrix}.$$

Dovendo eseguire il prodotto matrice-vettore $x^{(k+1)} = Ax^{(k)}$, e suddividendo anche il vettore $x^{(k+1)}$ in due vettori $x_N^{(k+1)}$ e $x_C^{(k+1)}$, possiamo riscrivere il prodotto appena visto

nella forma

$$\begin{bmatrix} x_N^{(k+1)} \\ x_C^{(k+1)} \end{bmatrix} = \begin{bmatrix} A_N \\ A_C \end{bmatrix} \begin{bmatrix} x_N^{(k)} \\ x_C^{(k)} \end{bmatrix}.$$

Per non ricalcolare i PageRank delle pagine già converse, non è necessario eseguire il calcolo $x_C^{(k+1)} = A_C x_N^{(k)} + A_C x_C^{(k)}$, ma è sufficiente impostare $x_C^{(k+1)} = x_C^{(k)}$.
Questa scelta ci permette dunque di ricondurre il risultato della $(k+1)$-sima iterazione del metodo alla risoluzione del seguente sistema:

$$\begin{cases} x_N^{(k+1)} = A_N x^{(k)}, \\ x_C^{(k+1)} = x_C^{(k)}. \end{cases}$$

Essendo A_N più piccola di A, il costo totale è minore di quello del prodotto $Ax^{(k)}$, e calerà ulteriormente al procedere delle iterazioni.

L'implementazione algoritmica di questo metodo richiede alcuni accorgimenti.

In primo luogo, è importante osservare che non è necessario ricostruire il blocco A_N ad ogni iterazione, ma è sufficiente modificare opportunamente la matrice A azzerandone il blocco A_C e conservare i PageRank già conversi in un nuovo vettore, evitando così di ricalcolarli.
Siano $A' = \begin{bmatrix} A_N \\ 0 \end{bmatrix}$ e $x_C'^{(k)} = \begin{bmatrix} 0 \\ x_C^{(k)} \end{bmatrix}$; possiamo esprimere un'iterazione dell'Adaptive PageRank come

$$x^{(k+1)} = A' x^{(k)} + x_C'^{(k)}.$$

La matrice A' ha le stesse dimensioni di A, ma è molto più sparsa, riducendo così il costo computazionale per ogni iterazione.

Inoltre, sebbene l'identificazione delle pagine converse non abbia un costo rilevante, il riordinamento della matrice A per ottenere il blocco appena descritto non è invece trascurabile; per evitare il riordinamento di A, è sufficiente costruire un vettore x'_C contenente gli indicatori delle pagine converse, e definire

$$A' = (A'_{i,j}) = \begin{cases} 0 & \text{se } i \in C, \\ a_{i,j} & \text{altrimenti.} \end{cases} \tag{1.2}$$

$$x_C'^{(k)} = (x_C'^{(k)})_i = \begin{cases} (x_C^{(k)})_i & \text{se } i \in C, \\ 0 & \text{altrimenti.} \end{cases} \tag{1.3}$$

L'ultima considerazione da fare è la frequenza con cui esaminiamo quali sono le pagine converse e aggiorniamo la matrice A e il vettore x_C. Eseguire questa operazione ad ogni ciclo rischierebbe infatti solo di diminuire l'efficienza totale di questo metodo.

Per questo motivo, dobbiamo stabilire una periodicità con cui questo controllo viene eseguito; se viene eseguito troppo frequentemente, l'algoritmo potrebbe subire dei rallentamenti dovuti al costo degli aggiornamenti, mentre se dovesse essere eseguito troppo raramente, l'algoritmo ricalcolerebbe più volte i PageRank di pagine già converse, non apportando miglioramenti notevoli all'efficienza generale del metodo.

Algoritmo 2 Filter-Based Adaptive PageRank

function FILTERAPR($A, x^{(0)}, v, \epsilon$)
  while $\delta > \epsilon$ **do**
    $x^{(k+1)} = A' x^{(k)} + x_C'^{(k)}$
    periodically
      $[N, C] = \text{detectConverged}(x^{(k)}, x^{(k+1)}, \epsilon)$ ▷ calcola i set di indici N, C
      $[A'] = \text{filter}(A, N, C)$ ▷ costruisce A' come nell'equazione (1.2)
      $[x'_C] = \text{filter}(x^{(k)}, C)$ ▷ costruisce x'_C come nell'equazione (1.3)
    $\delta = \|Ax^{(k)} - x^{(k)}\|_1$
  end while
end function

1.4 Modified Filter-Based Adaptive PageRank

Il metodo appena descritto può essere ulteriormente ottimizzato evitando di ricalcolare le componenti del PageRank delle pagine non ancora converse ricevute dalle pagine in C. Suddividiamo A in 4 sottomatrici:

$$A = \begin{bmatrix} A_{NN} & A_{CN} \\ A_{NC} & A_{CC} \end{bmatrix}$$

con $A_{NN} \in \mathbb{R}^{m \times m}$ contenente i collegamenti dalle pagine in N verso le pagine in N e $A_{CN} \in \mathbb{R}^{m \times (n-m)}$ contenente i collegamenti dalle pagine in C verso le pagine in N.

Utilizzando questa suddivisione di A, possiamo riscrivere il prodotto $x_N^{(k+1)} = Ax^{(k)}$ nella forma $x_N^{(k+1)} = A_{NN}x_N^{(k)} + A_{CN}x_C^{(k)}$.
Dal momento che eseguiamo il ricalcolo dei set N e C, e conseguentemente le modifiche alla matrice A, solo periodicamente, non sarà necessario ricalcolare il prodotto $A_{CN}x_C^{(k)}$ ad ogni iterazione, ma solo quando eseguiamo il controllo delle pagine converse.
Dunque, l'unico prodotto che deve essere eseguito ad ogni ciclo è il prodotto matrice-vettore $A_{NN}x_N^{(k)}$.

Analogamente alla prima versione dell'algoritmo, riordinare la matrice A in modo da ottenere i blocchi descritti è superfluo. Per ottenere il prodotto $A_{NN}x_N^{(k)}$ è sufficiente azzerare sia le righe che le colonne corrispondenti alle pagine già converse, e procedere come in precedenza.
Anche per il calcolo del prodotto $y = A_{CN}x_C^{(k)}$ non è necessario costruire la matrice A_{CN}, poiché è sufficiente moltiplicare il vettore $x_C^{(k)}$ per la matrice A, azzerando solo le righe corrispondenti alle pagine del set C, ossia

$$y = A'x'_C \tag{1.4}$$

Per costruire l'iterazione del metodo modificato nel modo appena descritto, dovremo quindi definire

$$A'' = (A''_{i,j}) = \begin{cases} 0 & \text{se } i,j \in C, \\ a_{i,j} & \text{altrimenti,} \end{cases} \tag{1.5}$$

per poi calcolare

$$x^{(k+1)} = A''x^{(k)} + y + x'_C.$$

Algoritmo 3 Modified Filter-Based Adaptive PageRank

function FILTERMAPR($A, x^{(0)}, y, v, \epsilon$)
  while $\delta > \epsilon$ **do**
    $x^{(k+1)} = A''x^{(k)} + y + x'_C$
    periodically
      $[N, C] = \text{detectConverged}(x^{(k)}, x^{(k+1)}, \epsilon)$
      $[A'', y] = \text{filterMOD}(A, N, C)$ ▷ costruisce A'' e y come in (1.5) e (1.4)
      $[x'_C] = \text{filter}(x^{(k)}, C)$ ▷ costruisce x'_C come nell'equazione (1.3)
    $\delta = \|Ax^{(k)} - x^{(k)}\|_1$
  end while
end function

Capitolo 2

Metodi randomizzati

In questo capitolo parleremo del modo in cui dei metodi randomizzati potrebbero essere utilizzati per la risoluzione del problema di PageRank, e cercheremo di individuare in quali situazioni questo tipo di metodi può essere più efficace dei metodi standard e adattivi già affrontati.

2.1 Approssimazione di Range(A)

In questo paragrafo studieremo una tecnica che ci permetterà di costruire una matrice Q che cattura una buona parte dell'azione di una matrice data $A \in \mathbb{R}^{m\times n}$.

Sia $A \in \mathbb{R}^{m\times n}$. Supponiamo di dover cercare una base ortonormale di k vettori, con $k < n$, che generi lo spazio vettoriale Range(A). Generiamo un vettore casuale gaussiano $\omega \in \mathbb{R}^n$, ed eseguiamo il prodotto $y = A\omega$; il vettore y così ottenuto è un vettore casuale contenuto in Range(A). Ripetendo lo stesso procedimento k volte, generando ogni volta un nuovo vettore gaussiano ω, otterremo un insieme di campioni

$$y^{(i)} = A\omega^{(i)}, \quad i = 1, 2, \ldots, k.$$

Con questi vettori, formiamo dunque la matrice campione $Y = [y^{(1)}, \ldots, y^{(k)}]$. La scelta casuale dei vettori $\{\omega^{(1)}, \ldots, \omega^{(k)}\}$ e il fatto che sia probabile che il rango di A non sia esattamente k, rendono molto probabile che le colonne di Y siano linearmente dipendenti; per questo motivo non è sufficiente normalizzare i vettori ottenuti per ottenere la base cercata, ma dovremo eseguire una fattorizzazione QR della matrice Y.
Così facendo, le colonne di Q costituiranno la base ortonormale cercata.

Sebbene il nostro scopo sia quello di costruire una base di k vettori ortonormali, spesso è necessario estrarre dei vettori campione y aggiuntivi; infatti, sia $A = B + E$, con $B \in \mathbb{R}^{m\times n}$ e di rango k, contenente l'informazione cercata, ed $E \in \mathbb{R}^{m\times n}$ una perturbazione. Lo scopo è quello di riuscire ad ottenere una base di k vettori il cui spazio generato approssimi bene Range(B). Costruendo la base $\{y^{(1)}, \ldots, y^{(k)}\}$ nel modo appena descritto, si osserva che i vettori di tale base sono nella forma

$$y^{(i)} = A\omega^{(i)} = B\omega^{(i)} + E\omega^{(i)}, \quad i = 1, 2, \ldots, k,$$

ossia si osserva che la matrice E condiziona i vettori $y^{(i)}$, rischiando che essi non appartengano allo spazio vettoriale Range(B), impedendo dunque alla base calcolata di

costruire lo spazio cercato. Per impedire che ciò accada, è utile stabilire un piccolo parametro p, detto *oversampling*, e costruire invece la base $\{y^{(1)}, \dots, y^{(k+p)}\}$, sempre nel modo appena visto; in questo modo, lo spazio generato da tale base ha più probabilità di approssimare Range(B).
La scelta dei parametri k e p dipende fortemente dalla natura del problema; per l'utilizzo di questi metodi che esploreremo in questo capitolo, sarà però sufficiente scegliere $k, p \sim 5, 10$. D'ora in avanti, poniamo $\ell = k + p$.

Algoritmo 4 Approssimazione randomizzata di Range(A)

1: **function** $[Q] =$ APPROXIMATION(A, ℓ)
2: $\quad \Omega = \text{randn}(n, \ell)$
3: $\quad Y = A\Omega$
4: $\quad [Q, R] = \text{qr}(Y)$
5: **end function**

Vediamo ora come verificare se Range(Q) sia una buona approssimazione di Range(A). Per farlo, studieremo come QQ^TA approssima A; infatti, eseguendo il primo prodotto Q^TA, proiettiamo lo spazio generato dalle colonne di A su Range(Q), ed eseguendo poi il secondo prodotto $Q(Q^TA)$, riportiamo questa proiezione in $\mathbb{R}^n$.
Se Range(Q) = Range(A), allora tutte le colonne di A si potrebbero scrivere come combinazione lineare delle colonne di Q, rendendo quindi Q^TA un cambiamento di base contenente i coefficienti di ogni colonna di A nella base di Range(Q). Moltiplicando questa matrice a sinistra per Q, si recupererebbe esattamente la matrice A di partenza, e dunque $QQ^TA = A$.
Tuttavia, se Range(Q) non coincidesse con Range(A), come nel caso in cui alcune colonne di A siano ortogonali alle colonne di Q, si perderebbero tali informazioni in Q^TA. Questa perdita di informazioni impedisce di recuperare completamente lo spazio di partenza Range(A) in $\mathbb{R}^n$, e di conseguenza QQ^TA non sarebbe più una buona approssimazione di A.

Per vedere quindi se Range(Q) è una buona approssimazione di Range(A), studiamo l'errore $\|A - QQ^TA\|_2 = \|(I - QQ^T)A\|_2$, utilizzando il seguente risultato.

Teorema 2.1. *[2] Siano $B \in \mathbb{R}^{m\times n}$. Fissiamo un numero intero positivo ℓ, e un numero reale $\alpha > 1$. Costruiamo poi una famiglia indipendente di vettori gaussiani standard $\{\omega^{(i)}, i = 1, 2, \dots, \ell\}$. Allora*

$$\|B\| \leq \alpha\sqrt{\frac{2}{\pi}} \max_{i=1,\dots,\ell} \|B\omega^{(i)}\|,$$

con probabilità $1 - \alpha^{-\ell}$.

Ponendo $B = (I - QQ^T)A$ e, ad esempio, $\alpha = 10$, otteniamo dunque un maggiorante dell'errore, in quanto

$$\|(I - QQ^T)A\| \leq 10\sqrt{\frac{2}{\pi}} \max_{i=1,\dots,\ell} \|(I - QQ^T)A\omega^{(i)}\|,$$

con probabilità $1 - 10^{-\ell}$.

Si osservi anche che per alti valori di α ed ℓ, la probabilità che la stima dell'errore sia corretta aumenta, ma anche il maggiorante aumenterebbe sensibilmente. Una scelta generale di α ed ℓ che rende la stima dell'errore accurata ed evita di aumentare troppo il maggiorante è, ad esempio, $\alpha, \ell = 10$.

2.2 Metodi di proiezione

Molti algoritmi per la risoluzione dei problemi agli autovalori contengono al proprio interno dei metodi di proiezione. Un metodo di proiezione consiste nell'approssimazione dell'esatto autovettore u di una matrice A tramite un autovettore $\tilde{u}$ che appartiene a un sottospazio K di $\mathbb{R}^n$, imponendo che il vettore residuo di $\tilde{u}$ sia ortogonale ad un secondo sottospazio L di $\mathbb{R}^n$ scelto.

Definizione 2.1. K è il *sottospazio di approssimazione*, o *sottospazio destro*, mentre L è detto *sottospazio sinistro*.

I metodi di proiezione si dividono in due grandi classi: i metodi di proiezione ortogonali, in cui scegliamo $K = L$, e i metodi di proiezione obliqui, in cui il sottospazio destro K e il sottospazio sinistro L sono distinti. All'interno di questa tesi, faremo ricorso solo ai metodi di proiezione ortogonali.

Siano dunque $A \in \mathbb{R}^{n\times n}$ e K un sottospazio vettoriale di $\mathbb{R}^n$. Cerchiamo un'autocoppia $(\tilde{\lambda}, \tilde{u})$, con $\tilde{u} \in K$, che approssimi l'autocoppia dominante di A, e che soddisfi l'equazione

$$\langle A\tilde{u} - \tilde{\lambda}\tilde{u}, v\rangle = 0, \ \forall v \in K. \tag{2.1}$$

Definizione 2.2. La condizione $\langle A\tilde{u} - \tilde{\lambda}\tilde{u}, v\rangle = 0, \ \forall v \in K$, è nota come *condizione di Galerkin*.

Supponiamo ora di conoscere una base ortonormale $\{v^{(1)}, \dots, v^{(\ell)}\}$ di K, e costruiamo con essa la matrice $V = \left[v^{(1)}, \dots, v^{(\ell)}\right]$; essendo $\tilde{u}$ un vettore appartenente allo spazio K, esiste $y \in \mathbb{R}^\ell$ tale che $\tilde{u} = Vy$.
Eseguendo questa sostituzione nella condizione (2.1), e imponendo la condizione di ortogonalità sulla base $\{v^{(1)}, \dots, v^{(\ell)}\}$ di K precedentemente considerata, otteniamo che $\tilde{\lambda}$ e y devono soddisfare la condizione

$$\langle AVy - \tilde{\lambda}Vy, v^{(j)}\rangle = 0, \quad \forall j = 1, \dots, \ell, \tag{2.2}$$

che è equivalente a richiedere che

$$V^*(AVy - \tilde{\lambda}Vy) = 0 \iff V^*AVy = \tilde{\lambda}y. \tag{2.3}$$

Ponendo $B_\ell = V^*AV \in \mathbb{R}^{\ell\times\ell}$, il nostro problema ora è calcolare y tale che $B_\ell y = \tilde{\lambda}y$, ossia dobbiamo trovare la soluzione di un problema agli autovalori proiettato, e potremo ottenere l'autovettore approssimato $\tilde{u}$ eseguendo il prodotto $\tilde{u} = Vy$.

Possiamo costruire una formulazione equivalente del problema, riconducendoci allo studio di un diverso problema agli autovalori per il calcolo dell'approssimazione $\tilde{u}$, considerando un proiettore ortogonale P_K sullo spazio K, ed esprimendo la condizione (2.1) nel seguente modo:

$$P_K(A\tilde{u} - \tilde{\lambda}\tilde{u}) = 0. \tag{2.4}$$

Dal momento che $\tilde{u}$ è contenuto nello spazio K, vale $P_K\tilde{u} = \tilde{u}$, e possiamo dunque esprimere l'equazione come

$$P_K A\tilde{u} = \tilde{\lambda}\tilde{u}, \quad \text{con } \tilde{u} \in K, \tag{2.5}$$

e per lo stesso motivo, anche

$$P_K A P_K \tilde{u} = \tilde{\lambda}\tilde{u}, \quad \text{con } \tilde{u} \in K. \tag{2.6}$$

Ponendo dunque $A_n = P_K A P_K$, per calcolare la soluzione approssimata ci siamo ricondotti a un problema agli autovalori di una trasformazione lineare di A.

Questa riformulazione ci permette di enunciare alcuni utili risultati per studiare la qualità dell'approssimazione così calcolata

Proposizione 2.1. *Se il sottospazio K è invariante per l'applicazione A, ogni autocoppia approssimata calcolata tramite il metodo di proiezione ortogonale è esatta.*

Dimostrazione. Sia $(\tilde{\lambda}, \tilde{u})$ un'autocoppia approssimata, allora vale la condizione (2.4), e dal momento che K è invariante per A, allora $P_K A\tilde{u} = A\tilde{u}$, e dunque $A\tilde{u} = \tilde{\lambda}\tilde{u}$, ossia l'autocoppia $(\tilde{\lambda}, \tilde{u})$ è esatta. □

Questa proposizione che ci fornisce un'importante risultato nel caso particolare in cui lo spazio K sia invariante per A, garantendoci la correttezza della soluzione approssimata. In generale però, per studiare l'accuratezza dei metodi di proiezione possiamo esaminare altre quantità. Un primo valore che possiamo considerare è la distanza tra l'autovettore esatto u e la sua proiezione sul sottospazio K, ossia $\|(I - P_K)u\|_2$; se questa quantità non è piccola, è immediato concludere che applicando un metodo di proiezione scegliendo K come sottospazio di approssimazione, non potremo ottenere una buona soluzione, in quanto $\|\tilde{u} - u\|_2 \leq \|(I - P_K)u\|_2$.

Una quantità più specifica che è utile studiare per capire se $(\tilde{\lambda}, \tilde{u})$ è una buona approssimazione di (λ, u) è il residuo dell'autocoppia esatta rispetto alla trasformazione lineare A_n, che possiamo maggiorare efficacemente tramite il seguente teorema.

Teorema 2.2. *Siano $\gamma = \|P_K A(I - P_K)\|_2$, (λ, u) autocoppia di A e P_K proiettore del sottospazio K. Allora le norme dei residui delle coppie $\lambda, P_K u$ e λ, u soddisfano le seguenti disuguaglianze:*

$$\|(A_n - \lambda I)P_k u\|_2 \leq \gamma \|(I - P_k)u\|_2, \tag{2.7}$$

$$\|(A_n - \lambda I)u\|_2 \leq \sqrt{\lambda^2 + \gamma^2}\|(I - P_k)u\|_2. \tag{2.8}$$

Dimostrazione. Per dimostrare la prima disuguaglianza, sfruttiamo le definizioni di A_n e di proiettore P_k.

$$\begin{aligned}\|(A_n - \lambda I)P_k u\|_2 &= \|(P_k A P_k - \lambda I)P_k u\|_2 = \|P_k A P_k u - \lambda P_k^2 u\|_2 \\ &= \|P_k(AP_k - \lambda I)P_k u\|_2 = \|P_k(A - \lambda I)(u - (I - P_k)u)\|_2 \\ &= \|P_k(A - \lambda I)(I - P_k)u\|_2 = \|P_k(A - \lambda I)(I - P_k)(I - P_k)u\|_2 \\ &\leq \gamma \|(I - P_k)u\|_2.\end{aligned}$$

Per dimostrare la seconda disuguaglianza, utilizziamo la proprietà dei proiettori per cui $u = P_k u + (I - P_k)u$, e la prima disuguaglianza:

$$(A_n - \lambda I)u = (A_n - \lambda I)P_k u + (A_n - \lambda I)(I - P_k)u = (A_n - \lambda I)P_k u - \lambda(I - P_k)u.$$

Dal momento che i due vettori della scomposizione sono ortogonali, abbiamo che

$$\|(A_n - \lambda I)u\|_2^2 = \|(A_n - \lambda I)P_k u\|_2^2 + \|\lambda(I - P_k)u\|_2^2 \leq \gamma^2 \|(I - P_k)u\|_2^2 + \lambda^2 \|(I - P_k)u\|_2^2,$$

e dunque

$$\|(A_n - \lambda I)u\|_2 = \sqrt{\gamma^2 + \lambda^2}\, \|(I - P_k)u\|_2.$$

□

2.3 Metodi di proiezione randomizzati

Vediamo ora come applicare i metodi appena descritti sfruttando l'approssimazione randomizzata di Range(A) calcolata nel paragrafo 2.1.

Avendo già a disposizione una base ortonormale $\{q^{(1)}, \ldots, q^{(\ell)}\}$ che dovrebbe approssimare lo spazio Range(A), sarà sufficiente costruire la matrice V utilizzando questa base impostando $V = Q$, e conseguentemente il sottospazio di approssimazione sarà $K = \text{Range}(Q)$.
La matrice $B_\ell \in \mathbb{R}^{\ell \times \ell}$ sarà dunque data dal prodotto $B_\ell = Q^* A Q$, e il metodo per calcolare l'approssimazione $\tilde{u}$ del vettore di PageRank, consiste nel seguente algoritmo.

Algoritmo 5 Calcolo di $\tilde{u}$ tramite un problema agli autovalori proiettato

function $[\tilde{u}]$ = PAGERANK_APPROXIMATION$(A, \ell, y^{(0)}, \text{tol})$
  $Q = \text{Approximation}(A, \ell)$ ▷ Algoritmo 4
  $B_\ell = Q^* A Q$
  $y = \text{Standard_Power_Method}(B_\ell, y^{(0)}, \text{tol})$ ▷ $y^{(0)} \in \mathbb{R}^\ell$
  $\tilde{u} = Qy$
end function

È importante soffermarsi sul modo in cui calcoliamo il vettore y: con le ipotesi presentate fino ad ora, non possiamo avere la certezza che l'autovettore dominante di B_ℓ sia quello che ci permetterà di approssimare al meglio il vettore di PageRank tramite

$\tilde{u} = Qy$.
Essendo B_ℓ una matrice molto piccola, sarebbe possibile calcolare direttamente tutto il suo spettro e tutti i suoi autovettori. Tuttavia, considerando anche che tale matrice è ottenuta partendo da una matrice generata randomicamente, non possiamo avere la certezza che l'autovalore di B_ℓ che meglio approssima l'autovalore dominante $\lambda = 1$ di A sia il suo autovalore dominante, e non possiamo nemmeno sapere se il vettore ottimale y sia l'autovettore dominante, o l'autovettore associato all'autovalore di B_ℓ che meglio approssima 1.

Garantendo che anche B_ℓ rispetti le ipotesi del teorema di Perron-Frobenius, avremmo la certezza che anche il suo autovalore dominante sia $\tilde{\lambda} = 1$, e di conseguenza $\tilde{u}$ dovrebbe essere la migliore approssimazione di I che possiamo ottenere in Range(Q).
Sebbene sia possibile modificare direttamente la matrice B_ℓ, la soluzione più stabile si ottiene imponendo delle condizioni sulla matrice Q affinché la matrice $B_\ell = Q^T AQ$ sia stocastica per colonne e irriducibile. Se B_ℓ deve essere stocastica per colonne, allora deve valere $B_\ell^T \mathbf{1} = Q^T A^T Q\mathbf{1} = \mathbf{1}$. Sapendo già che A è stocastica per colonne, cioè $A^T\mathbf{1} = \mathbf{1}$, dunque questa uguaglianza può essere soddisfatta imponendo che $\mathbf{1}$ appartenga a Range(Q), e lo facciamo aggiungendo $\mathbf{1}$ alle colonne di Q, e procedendo a ortogonalizzare la matrice.
La condizione di irriducibilità è invece generalmente più semplice da ottenere; è ad esempio sufficiente che la matrice non abbia alcun coefficiente nullo, perché in tal caso il grafo associato a B_ℓ sarebbe fortemente connesso, e otterremo l'irriducibilità di B_ℓ dal teorema (0.1).In generale, non abbiamo però dimostrato che B_ℓ è sempre irriducibile, ma lo è sempre stata nei nostri esperimenti.

La riformulazione precedentemente proposta ci consente di stimare quanto la soluzione esatta u sia contenuta nello spazio Range(Q): consideriamo come proiettore sullo spazio K la matrice QQ^*, e riconduciamoci alla risoluzione del problema agli autovalori (2.6),

$$A_n\tilde{u} = QQ^T AQQ^T\tilde{u} = \tilde{\lambda}\tilde{u}.$$

Moltiplicando Q^T a sinistra di entrambi i membri dell'equazione, otteniamo

$$Q^T AQQ^T\tilde{u} = \tilde{\lambda}Q^T\tilde{u},$$

e ponendo $y = Q^T\tilde{u}$, possiamo ricondurci proprio all'equazione (2.3)

$$Q^T AQy = B_\ell y = \tilde{\lambda}y,$$

e dunque $\tilde{u} = Qy = QQ^T\tilde{u}$. Se al posto di $\tilde{u}$ considerassi la soluzione esatta u, questa equazione sarebbe vera se e solo se u è contenuto in Range(Q); studiare la differenza $u - QQ^T u$ ci permette dunque di capire quanto la soluzione esatta sia distante da Range(Q).
Un'ultima condizione che dovremmo imporre su B_ℓ, sarebbe che essa sia a componenti non negative: un modo efficace per assicurare ciò, è imporre che Q stessa sia a termini non negativi, in quanto sappiamo già che questo vale per A. Per ottenere dunque una matrice Q a componenti non negative, sarebbe sufficiente imporre che la matrice randomizzata Ω sia a componenti non negative, così che anche $A\Omega$ lo sia, e dunque anche la Q della sua decomposizione QR, ma non abbiamo scelto di implementare questa strategia, preferendo utilizzare la distribuzione `randn` per costruire Ω.

Capitolo 3

Esperimenti numerici

In questo paragrafo analizziamo i risultati ottenuti applicando i metodi adattivi e randomizzati ad alcune matrici disponibili sul sito

`https://www.cise.ufl.edu/research/sparse/matrices/list_by_id.html`

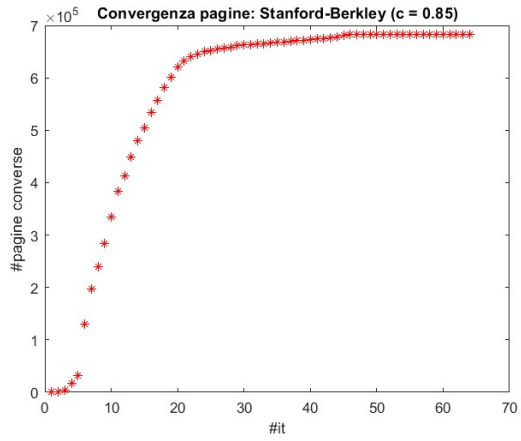
Ho eseguito i test su quattro matrici di diverse dimensioni:

- **Stanford-Berkeley**, $n = 683446$, `SNAP/web-BerkStan`;
- **web-Google**, $n = 916428$, `SNAP/web-Google`;
- **Stanford**, $n = 281903$, `SNAP/web-Standford`;
- **web-Notredame**, $n = 325729$, `SNAP/web-NotreDame`.

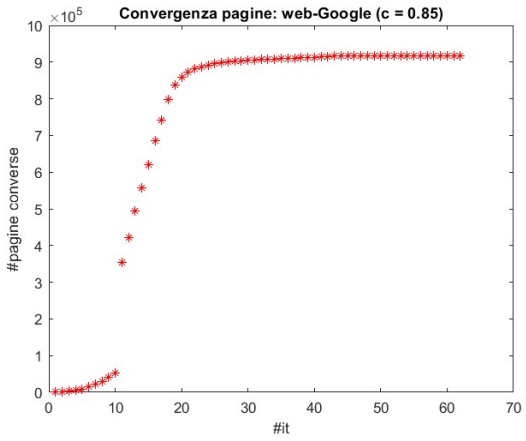
3.1 Esperimenti su Adaptive Pagerank

Osserviamo ora i risultati dei metodi di Adaptive Pagerank e della sua versione modificata, paragonati ai risultati ottenuti tramite il metodo delle potenze classico.

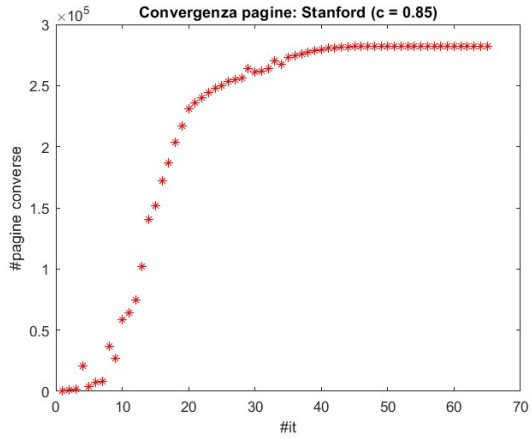
Cominciamo osservando la velocità di convergenza delle pagine dei dataset. Ho scelto $c = 0.85$ come parametro di condizionamento per rendere la matrice irriducibile, e ho considerato la condizione (1.1) come criterio di convergenza dei Pagerank delle singole pagine.
I grafici riportati di seguito mostrano la quantità di Pagerank conversi ad ogni iterazione del metodo delle potenze standard.



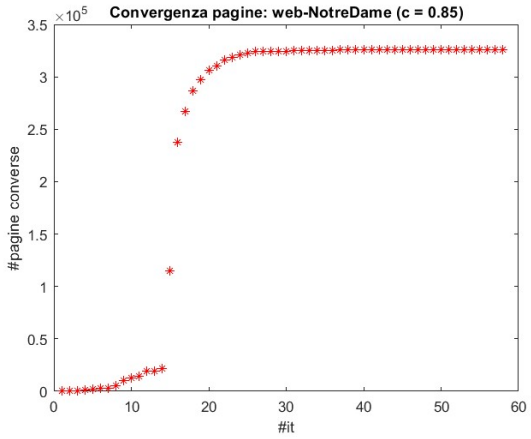
(a)



(b)



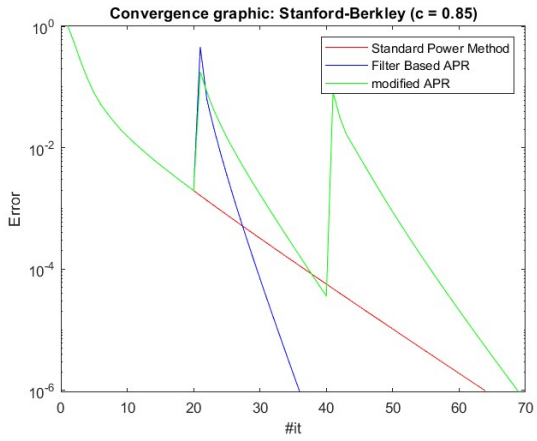
(c)



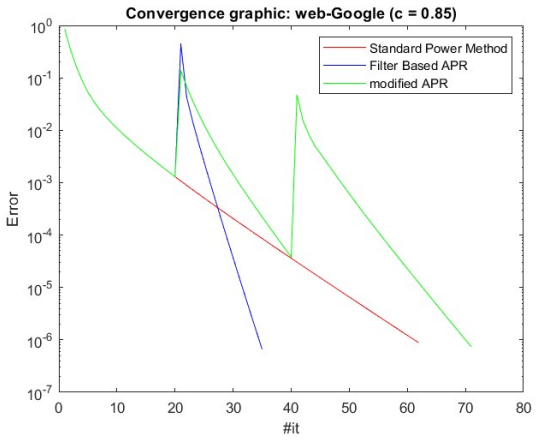
(d)

Tutti i dataset utilizzati presentano delle ottime condizioni per applicare i metodi adattivi, in quanto la quasi totalità dei Pagerank converge in meno di 20 iterazioni. Le pagine restanti necessitano invece di altre 40/50 iterazioni per riuscire a convergere ai loro pagerank effettivi.
Impostando dunque il controllo dei PageRank conversi ogni 20 iterazioni e la tolleranza sul residuo $\delta = \|Ax^{(k)} - x^{(k)}\|_1 < 10^{-6}$, i tempi necessari per la convergenza sono diminuiti, anche se questo non corrisponde necessariamente a un numero inferiore di iterazioni necessarie.

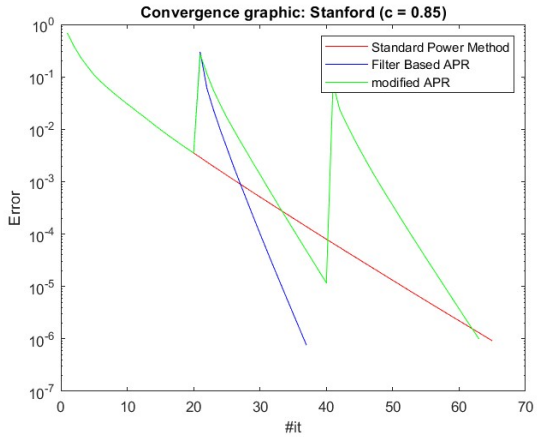
I grafici di seguito riportati rappresentano l'andamento dei residui nel corso delle varie iterazioni dei metodi adattivi, paragonati a quello del metodo delle potenze.



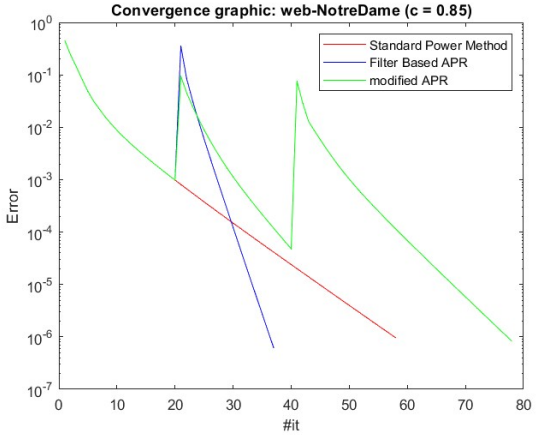
(e)



(f)



(g)



(h)

Inizialmente i residui coincidono per tutti i metodi, in quanto nelle prime 20 iterazioni il procedimento fra i metodi rimane invariato, mentre ogni volta che il controllo sui Pagerank viene eseguito assistiamo ad un iniziale aumento del residuo, che cala però drasticamente nelle iterazioni successive. Impostando il controllo dei Pagerank conversi ogni 20 iterazioni, per la prima versione dell'algoritmo è necessario eseguire il controllo soltanto una volta, mentre la versione modificata esegue il controllo sempre tre volte.

Riportiamo di seguito i dati sul tempo di esecuzione, il numero di iterazioni, e i residui in norma 2 delle soluzioni ottenute tramite i diversi metodi.

Stanford-Berkley

	Tempo	**Iterazioni**	$\|\|\mathbf{AI} - \mathbf{I}\|\|_\mathbf{2}$
Potenze Standard	2.1588	64	2.598055e-08
FilterAPR	0.9195	36	1.462591e-02
ModFilterAPR	1.5091	69	4.082092e-03

web-Google

	Tempo	**Iterazioni**	$\|\|\mathbf{AI}-\mathbf{I}\|\|_\mathbf{2}$
Potenze Standard	4.3536	62	1.994552e-08
FilterAPR	1.9731	35	3.113437e-03
ModFilterAPR	3.2167	71	6.918691e-04

Stanford

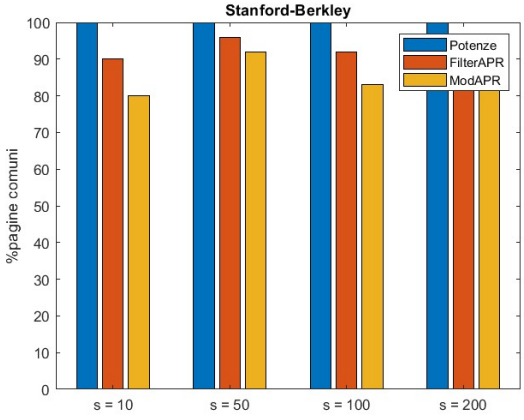
	Tempo	**Iterazioni**	$\|\|\mathbf{AI}-\mathbf{I}\|\|_\mathbf{2}$
Potenze Standard	1.2903	65	9.113734e-08
FilterAPR	0.6939	37	2.194177e-03
ModFilterAPR	1.0014	63	2.295460e-03

web-NotreDame

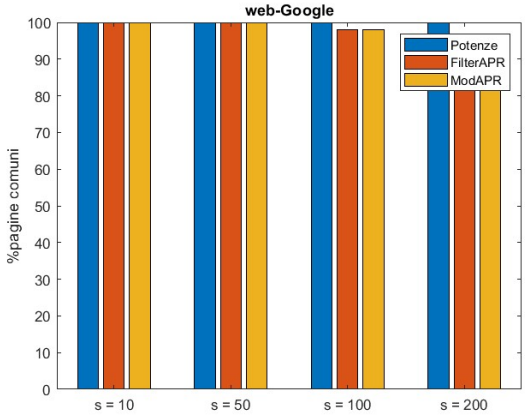
	Tempo	**Iterazioni**	$\|\|\mathbf{AI}-\mathbf{I}\|\|_\mathbf{2}$
Potenze Standard	0.7187	58	3.321039e-08
FilterAPR	0.4352	37	6.226379e-03
ModFilterAPR	0.7029	78	4.192628e-03

Si osserva che il Filter-Based adaptive pagerank velocizza la computazione del pagerank di circa il 50%, eseguendo in media il 58% delle iterazioni rispetto al metodo delle potenze standard. La versione modificata rappresenta invece un miglioramento rispetto all'algoritmo iniziale, ma non riesce a superare la velocità della prima versione di adaptive pagerank sui dataset proposti, ottenendo un incremento della velocità di esecuzione al massimo del 25%.

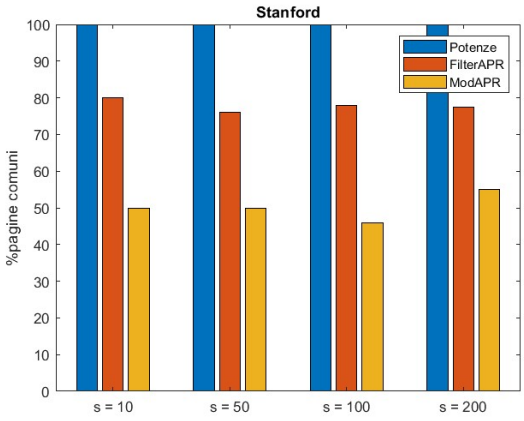
Il residuo del vettore di Pagerank $\|AI - I\|_2$ ottenuto tramite i metodi di Adaptive Pagerank si attesta ad ordini di grandezza di $10^{-2}/10^{-4}$, fornendo dunque una soluzione sensibilmente più distante dall'autovettore dominante della matrice A, ma che è in grado di fornire comunque una buona approssimazione dei pagerank più alti contenuti nel dataset. Nei seguenti grafici riportiamo la percentuale di pagine che i vari metodi sono riusciti a raccogliere fra le prime 10, 50, 100 e 200 pagine più importanti di ciascun dataset.



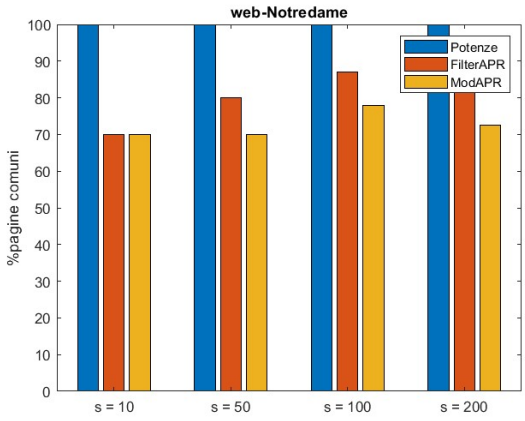
(i)



(j)



(k)



(l)

Il metodo delle potenze riesce a calcolare correttamente tutti i pagerank più alti delle prime 200 pagine in tutti i dataset proposti, ma è anche il metodo dal tempo di esecuzione maggiore.
Le versioni di Adaptive Pagerank hanno invece prestazioni diverse su tutti i dataset: questi metodi restituiscono ottimi risultati sul dataset Stanford-Berkley, in cui riescono a catturare fra l'80% e il 90% delle pagine più importanti in ogni insieme di pagine analizzato (i), ma i risultati sono ancora migliori sulla matrice web-Google, in cui riescono a identificare correttamente quasi tutte le pagine contenute in ogni insieme preso in considerazione (j).
Sui dataset di dimensione minore tali metodi presentano comunque dei risultati discreti: la prima versione di Adaptive Pagerank riesce a identificare in media l'80% delle pagine di tutti gli insiemi analizzati su entrambi i dataset, mentre la versione modificata cattura in media il 75% delle pagine della matrice web-Notredame (l), e solo il 50% delle pagine della matrice Stanford (k).

3.2 Esperimenti sui metodi randomizzati

Testiamo ora l'efficienza del metodo randomizzato sui dataset proposti al variare della dimensione ℓ scelta per approssimare lo spazio delle colonne di A.
Come termine di paragone, utilizziamo il risultato fornito dalla funzione `eigs` di Matlab.

Stanford-Berkley

	Tempo	$\|AI - I\|_2$
eigs	10.6253	4.210450e-14
$\ell = 10$	0.4796	1.7214
$\ell = 20$	1.0718	1.0907
$\ell = 30$	1.7371	1.2788
$\ell = 40$	1.7332	0.9837
$\ell = 50$	2.3480	0.9256
$\ell = 60$	2.7308	0.8447
$\ell = 70$	3.4458	0.8935
$\ell = 80$	3.9089	0.8481
$\ell = 90$	4.5185	0.7873
$\ell = 100$	5.1844	0.7305

web-Google

	Tempo	$\|AI - I\|_2$
eigs	19.4271	5.195309e-15
$\ell = 10$	0.9541	5.1407
$\ell = 20$	2.4742	3.6009
$\ell = 30$	3.8247	4.3806
$\ell = 40$	3.7086	3.2620
$\ell = 50$	4.7821	3.1273
$\ell = 60$	5.9329	2.9229
$\ell = 70$	7.0808	2.7670
$\ell = 80$	8.0178	2.5722
$\ell = 90$	9.1679	2.3339
$\ell = 100$	10.9737	2.3929

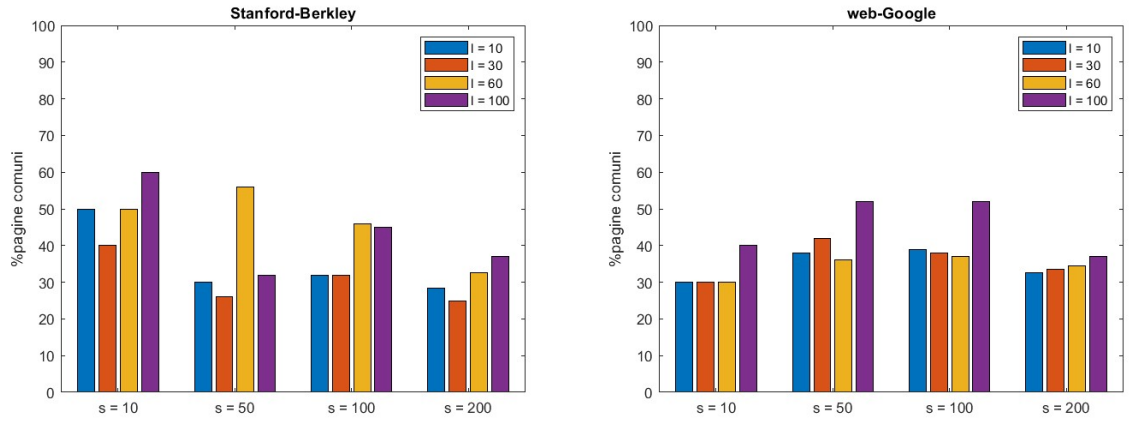
Stanford

	Tempo	$\|AI - I\|_2$
eigs	5.0263	5.767103e-15
$\ell = 10$	0.3004	2.1978
$\ell = 20$	0.6986	2.1804
$\ell = 30$	1.2317	2.1522
$\ell = 40$	1.3132	2.1362
$\ell = 50$	1.6177	2.0797
$\ell = 60$	1.8692	2.0689
$\ell = 70$	2.1674	2.0383
$\ell = 80$	2.5258	2.0377
$\ell = 90$	2.8992	2.0186
$\ell = 100$	3.2447	1.9752

web-NotreDame

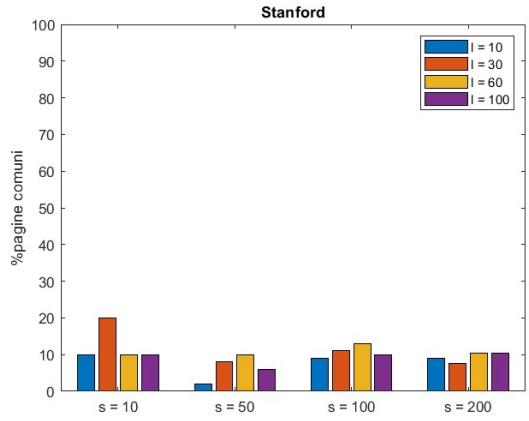
	Tempo	$\|AI - I\|_2$
eigs	4.0292	1.053676e-14
$\ell = 10$	0.2103	1.8011
$\ell = 20$	0.4601	1.4186
$\ell = 30$	0.7628	1.3388
$\ell = 40$	0.7860	1.1388
$\ell = 50$	1.0657	1.2386
$\ell = 60$	1.2402	1.0453
$\ell = 70$	1.4740	1.0093
$\ell = 80$	1.7215	1.0275
$\ell = 90$	1.9934	0.9399
$\ell = 100$	2.2751	1.0066

La velocità di computazione del metodo è notevolmente migliorata anche per alti valori di ℓ, raggiungendo una diminuzione del tempo di esecuzione in media fino al $\sim 56\%$ sui dataset analizzati. Il residuo $\|AI - I\|_2$ della soluzione è alto, attestandosi su ordini di grandezza di $10^0/10^{-1}$, dunque il vettore trovato non è in generale una buona approssimazione dell'effettivo vettore di Pagerank, ma la soluzione proposta riesce spesso a catturare le pagine più importanti contenute nel dataset.

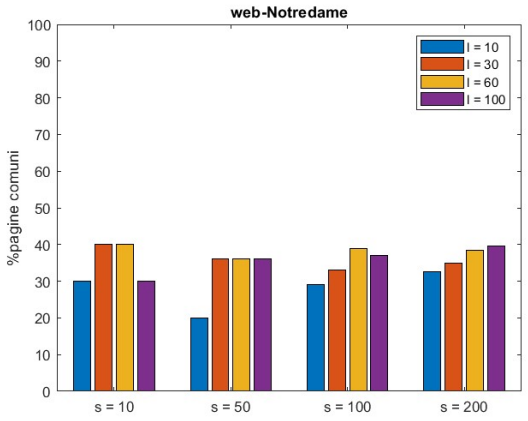


(a)

(b)



(c)



(d)

Eseguendo un controllo sulle prime 50 pagine fornite dal metodo randomizzato su ciascuno dei dataset proposti, si osserva che circa il 37% di esse si ritrova anche nelle prime 50 pagine fornite dalla funzione eigs, fatta eccezione per il dataset Stanford, la cui percentuale di pagine in comune raggiunge quasi il 10%. Tale percentuale non incrementa sensibilmente all'aumentare del valore di ℓ; può anzi capitare che, a causa della natura randomizzata del metodo, il numero di pagine comuni cali lievemente rispetto a valori di ℓ minori. All'aumentare del numero di pagine su cui si esegue un controllo, queste percentuali calano, in quanto i vettori di PageRank ottenuti tramite questi metodi hanno dei residui molto alti, al contrario della soluzione fornita da eigs. Per esempio, sulle prime 100 pagine, circa il 30% di esse si ritrova anche nelle prime 100 pagine fornite dalla funzione eigs, e la percentuale rimane invariata eseguendo un controllo sulle prime 200 pagine.
È interessante osservare che il dataset Stanford non gode di queste ultime percentuali di pagine in comune, ma la percentuale inziale rimane stabile anche all'aumentare del numero di pagine analizzata: sebbene la qualità della soluzione approssimata per tale dataset non differisca particolarmente da quella delle soluzioni per gli altri dataset, il vettore ottenuto per questo dataset è sensibilmente peggiore nell'identificare le pagine più importanti.

Il vettore di PageRank ottenuto tramite i metodi randomizzati non rappresenta dunque una buona approssimazione generale del vettore di PageRank esatto, ma riesce spesso a cogliere, anche per bassi valori di ℓ, una quantità di informazioni sul dataset sufficiente per poter cominciare a ricercare una soluzione più precisa. Un buon uso di queste approssimazioni potrebbe essere quello di utilizzarle come iterati iniziali dei metodi delle potenze standard o di Adaptive PageRank, in modo da velocizzare ulteriormente l'esecuzione di questi algoritmi.

Conclusioni

In questa tesi ho presentato vari tipi di metodi per la risoluzione del problema di Pagerank, e ne ho analizzato l'efficienza e l'accuratezza su diversi dataset. Ho dapprima esposto il metodo classico per la risoluzione del problema, parlando successivamente di metodi adattivi, e soprattutto di metodi randomizzati, una nuova tipologia di metodi ancora in fase di sperimentazione e che si basano sull'approssimazione dello spazio generato dalle colonne della matrice di Hyperlink, permettendo di ricondurci alla risoluzione di un problema agli autovalori proiettato.

Ho infine implementato tutti i metodi descritti in macchina, studiandone il comportamento su diverse matrici e per diversi parametri: dagli esperimenti numerici è emerso come i metodi randomizzati non siano in grado, nemmeno per approssimazioni di grandi dimensioni, di fornire una valida approssimazione dell'intero vettore esatto di Pagerank, ma che riescano comunque a catturare una parte dei Pagerank più alti all'interno dei dataset. Nonostante ciò, questi metodi si sono rivelati molto efficienti paragonati ai metodi classici, motivo per cui i risultati ottenuti tramite essi possono rappresentare degli ulteriori strumenti per migliorare le prestazioni di altri tipi di metodi già verificati.

Bibliografia

[1] Sepandar Kamvar, Taher Haveliwala, Gene Golub, *Adaptive methods for the computation of PageRank*, Linear Algebra and its Applications, Vol. 386, 2004, pp. 51–65.

[2] N. Halko, P. G. Martinsson, J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decomposition*, SIAM Review, Vol. 53, No. 2, 2011, pp. 217–288.

[3] V. Simoncini, *Dispense per il corso di Matematica Computazionale*

[4] L. Grimetti, *Metodi numerici per il calcolo del Pagerank*, 2018, pp 15–20