Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

METODI DI ESTRAPOLAZIONE E ADATTIVI PER IL CALCOLO DEL PAGERANK

Tesi di Laurea in Matematica Computazionale

Relatore:
Chiar.ma Prof.ssa
VALERIA SIMONCINI

Presentata da:
CHIARA BASSI

Anno Accademico 2020-2021
Sessione IV · 28 ottobre 2021

Indice

Introduzione

In questa tesi viene studiato l'algoritmo di PageRank, un metodo per determinare il ranking, ovvero l'ordinamento, delle pagine web tramite l'assegnazione di pesi numerici.
Quando si compie una ricerca per esempio su Google, si vorrebbe avere una risposta pertinente nel minor tempo possibile e nei primi siti che compaiono, senza dover esaminare tutti i risultati.
Questa necessità era già presente negli anni '90 quando Internet cominciò ad avere sempre più documenti e perciò si avvertì l'esigenza di catalogarli. Larry Page e Sergey Brin, nel 1998, formularono l'algoritmo di PageRank che si basa sulla seguente strategia: l'importanza di una pagina web è determinata considerando il numero di links da e verso quella pagina. In particolare l'importanza di una pagina aumenta se ad essa sono collegate molte pagine a loro volta importanti. La verifica dell'importanza, inoltre, viene fatta in maniera meccanica, senza l'intervento umano.

Nel primo capitolo viene introdotto l'algoritmo del PageRank fornendo delle definizioni di base, fra cui quella di vettore di PageRank e di matrice di Hyperlink. Viene poi esposto come modificare la matrice di Hyperlink in modo tale che risulti stocastica per colonna e irriducibile. Infine viene sviluppato l'algoritmo del metodo delle potenze.
Nel secondo capitolo vengono introdotti tre metodi di estrapolazione che accelerano la convergenza del metodo delle potenze: estrapolazione di Aitken, estrapolazione epsilon ed estrapolazione quadratica. Per ogni metodo si approfondisce lo sviluppo dell'algoritmo, il costo computazionale e infine vengono riportati dei risultati sperimentali dove viene utilizzato il dataset "Stanford-Berkeley".
Nel terzo capitolo vengono trattati i metodi adattivi. L'idea alla base di questi metodi è che non tutte le pagine convergono al loro valore di PageRank in modo uniforme: molte pagine convergono velocemente mentre altre, che generalmente sono quelle con un pagerank più alto, convergono più lentamente. Anche per questi metodi vengono mostrati alcuni risultati sperimentali come il numero di iterazioni ed il tempo di convergenza.

Nel capitolo 4 viene proposta un'analisi sperimentale conclusiva dei metodi visti nella tesi, utilizzando diversi database. In particolare si ha un confronto fra tutti i metodi di estrapolazione ed un confronto fra l'estrapolazione ed i metodi adattivi.

Capitolo 1

Algoritmo del PageRank

L'algoritmo del PageRank si basa sull'idea di assegnare un peso numerico, detto pagerank, ad ogni pagina Web ed ha come scopo quello di fornire un ranking, ovvero una classifica, di tutte le pagine considerate basandosi sull'importanza che esse hanno.

La strategia adoperata da Google si basa sul presupposto che il numero dei links da e verso una pagina Web permettono di determinare l'importanza della pagina: il link da una pagina u a una pagina v permette a v di acquisire valore, in particolare l'importanza che viene conferita a v da u è direttamente proporzionale alla rilevanza di u e inversamente proporzionale al numero di links uscenti da u. Dato che il peso di u non è noto, determinare l'importanza di una pagina richiede l'utilizzo di metodi iterativi con punto fisso.

1.1 Introduzione

Il Web è un insieme vastissimo di documenti, grazie alla sua struttura ipertestuale è possibile definire il Web come un grafo diretto:

Definizione 1 (Grafo diretto)**.** Un grafo diretto G è una coppia di elementi (V,E) dove $V = \{n_1, \ldots, n_n\}$ è un insieme finito di elementi detti nodi (o vertici) del grafo ed $E = \{e_1, \ldots, e_n\} \subseteq V \times V$ è un sottoinsieme dell'insieme delle coppie dei nodi, i cui elementi si chiamano archi del grafo.

Due pagine P_i e P_j rappresentano due vertici e l'esistenza di un link dalla pagina P_i alla pagina P_j è individuato da un arco. Chiamiamo *outlinks di* P_i le pagine che vengono puntate da P_i e *inlinks di* P_i le pagine che hanno come outlinks P_i.

Secondo la strategia utilizzata da Page e Brin più sono importanti i collegamenti ad una pagina P_i più essa assume importanza. L'importanza della pagina P_i, che indichiamo con $I(P_i)$, è data da:

$$I(P_i) = \sum_{j \in B_{P_i}} \frac{I(P_j)}{deg(j)},$$

dove B_{P_i} è l'insieme degli inlinks di P_i e deg(j) è il numero di outlinks dalla pagina P_j.

Definiamo ora il vettore di PageRank $\vec{I}$ che contiene i pagerank di tutte le pagine.

Definizione 2 (Vettore di PageRank)**.** Il vettore di PageRank $\vec{I}$ è definito come $(I(P_i))_{i=1,..,n}$, dove n è il numero delle pagine totali considerate.

Definizione 3 (Matrice di Hyperlink)**.** La matrice H si dice di Hyperlink se descrive la transizione da P_i e P_j, $H = (h_{i,j})$ con

$$h_{i,j} = \begin{cases} \frac{1}{deg(j)} & \text{se } P_j \text{ punta a } P_i, \\ 0 & \text{altrimenti.} \end{cases}$$

Si ha quindi che le righe della matrice indicano gli inlinks mentre le colonne indicano gli outlinks. L'elemento $H^T_{i,j}$ indica la probabilità di passare dalla pagina P_j alla pagina P_i.

La definizione data per $I(P_i)$ si traduce in termini matriciali come: $\vec{I} = H\vec{I}$, cioè la matrice H ha autovettore $\vec{I}$ e autovalore 1. Per calcolare, quindi, il vettore di Pagerank $\vec{I}$ è sufficiente determinare l'autovettore relativo all'autovalore 1 della matrice H.

1.2 Modifiche della matrice di Hyperlink

Il vettore di PageRank, che soddisfa $\vec{I} = H\vec{I}$, può essere calcolato in modo iterativo utilizzando il metodo delle potenze

$$\vec{I}^{(k+1)} = H\vec{I}^{(k)}.$$

Tuttavia questa procedura iterativa presenta dei problemi:

1. potrebbero esistere pagine prive di outlinks,
2. la matrice H potrebbe non avere 1 come autovalore,

3. l'autovalore 1 potrebbe avere più di un autovettore associato.

Per superare questi limiti sono state apportate modifiche all'algoritmo originale.

Affinché la procedura iterativa sia ben definita un ipotetico utente non può rimanere bloccato in una pagina P_i a causa della mancanza di outlinks di P_i. Dunque nel grafo non devono esserci dei nodi che non portano a nessun altro nodo, tali nodi vengono chiamati dangling nodes. Questa condizione equivale ad imporre che H non abbia colonne nulle.

Definizione 4 (Matrice stocastica)**.** La matrice $A = (a_{i,j})$ di dimensioni $n \times n$ si dice stocastica se i suoi elementi sono non negativi e se la somma degli elementi su ogni riga (o su ogni colonna) è uguale a 1:

$$a_{i,j} \geq 0 \quad \sum_j a_{i,j} = 1 \quad i = 1, 2, .., n.$$

Si vuole, quindi, modificare la matrice H rendendola una matrice stocastica per colonna, sostituendo ogni colonna di zeri con una colonna costante.
Sia n il numero di nodi. Si definisca il vettore $\vec{v}$ come il vettore n-dimensionale colonna che rappresenta la distribuzione della probabilità uniforme fra tutti i nodi. $\vec{v}$ viene detto vettore di personalizzazione in quanto ridefinendolo, senza utilizzare la probabilità uniforme, si cambia arbitrariamente l'importanza di una pagina

$$\vec{v} = \left[\frac{1}{n}\right]_{n\times 1}.$$

Il vettore $\vec{d}$ è il vettore n-dimensionale colonna che identifica i nodi con nessun link uscente

$$d_i = \begin{cases} 1 & \text{per } deg(i) = 0, \\ 0 & \text{altrimenti.} \end{cases}$$

Sia ora D la matrice, il cui effetto è quello di modificare le probabilità di passare da una pagina all'altra in modo che, se il visitatore finisce su una pagina senza links in uscita, esso verrà reindirizzato, utilizzando la distribuzione data da $\vec{v}$, su un'altra pagina

$$D = \vec{d} \cdot \vec{v}^T.$$

Considerando quindi le osservazioni appena fatte possiamo definire la matrice H':

$$H' = H + D.$$

Si osservi che la matrice H' è una matrice stocastica per colonna e che soddisfa $(H')^T 1 = 1$, cioè H' ha autovalore sinistro 1. Quindi con questa modifica si risolve sia il problema dei dangling nodes sia si impone l'esistenza di 1 come autovalore.

Occorre, inoltre, imporre che la matrice H' sia irriducibile affinché si abbia l'unicità del vettore di PageRank, utilizzando quindi la teoria di Perron-Frobenius che segue.

Definizione 5 (Matrice riducibile)**.** La matrice $A \in \mathbb{R}^{n \times n}$ si dice riducibile se esiste una matrice di permutazione T tale che

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

con X,Z matrici quadrate.

Perciò, se la matrice H è riducibile si ha un grafo non connesso, ovvero un grafo in cui esiste almeno un nodo dove vi sono esclusivamente outlinks.

Definizione 6 (Grafo fortemente connesso)**.** Il grafo diretto di una matrice A si dice fortemente connesso se per ogni elemento nonzero $a_{i,j}$ c'è un percorso nel grafo che porta P_i a P_j.

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

Teorema 2 (di Perron-Frobenius, variante)**.** *Sia A irriducibile e stocastica per colonna. Allora l'autovalore dominante, $\lambda_1 = max|\lambda_i|$, è uguale a 1 ed esiste un unico vettore associato con componenti tutte positive. Inoltre, se A ha elementi tutti positivi, $|\lambda_i| < 1$, $i = 2, 3, .., n$.*

Si modifichi quindi H' come segue:
sia E la matrice che permette al visitatore di saltare in una pagina casuale, con probabilità (1-α), utilizzando la distribuzione data dal vettore $\vec{v}$

$$E = [1]_{n \times 1} \times \vec{v}^T.$$

Definiamo ora la matrice H'' che modifica in modo opportuno la matrice H' in modo tale da avere una matrice irriducibile:

$$H'' = \alpha H' + (1 - \alpha)E.$$

La matrice H'' è stocastica per colonna e irriducibile con elementi tutti positivi; inoltre, ha come primo autovalore 1 e l'unico vettore ad esso associato ha componenti tutte positive.

Il parametro α corrisponde alla probabilità che un navigatore passi da una pagina ad un'altra in maniera arbitraria. Quando questa probabilità è piccola ed il vettore $\vec{v}$ è uniforme il metodo delle potenze è efficiente e si ha convergenza veloce; tuttavia, in questo caso, anche pagine con un'importanza minore hanno un pagerank più elevato del loro valore effettivo, ovvero se $1-\alpha$ è grande allora per l'effetto di E le pagine tendono ad acquisire una probabilità uniforme, che si otterrebbe con $1-\alpha=1$.
Viceversa se α è grande allora si ha maggior attinenza alla struttura del Web, ottenendo però una convergenza più lenta.

Il seguente teorema mostra come cambiano gli autovalori della matrice H''.

Teorema 3. *Sia spec(A)=*$\{1,\lambda_2,..,\lambda_n\}$*. Allora P=* $\alpha A+(1-\alpha)\frac{1}{n}\vec{1}\vec{1}^T$ *è tale per cui spec(P)=*$\{1,\alpha\lambda_2,..,\alpha\lambda_n\}$.

Dimostrazione. Sia $\vec{u}=\frac{1}{\sqrt{n}}\vec{1}$.
Si osservi che $P^T\vec{u}=\alpha A^T\vec{u}+(1-\alpha)\frac{1}{n}\vec{1}\vec{1}^T\vec{u}=\vec{u}$.
Sia ora $U=\begin{bmatrix}\vec{u} & U_0\end{bmatrix}$ unitaria, allora $U^*AU=\begin{bmatrix}1 & 0\\ * & T\end{bmatrix}$.
Si ottiene quindi

$$U^*PU=\alpha\begin{bmatrix}1 & 0\\ * & T\end{bmatrix}+(1-\alpha)\frac{1}{n}ne_1\vec{u}^TU=\begin{bmatrix}1 & 0\\ * & \alpha T\end{bmatrix}$$

□

Il Teorema 3 afferma, quindi, che a prescindere della molteplicità dell'autovalore 1 associato alla matrice H, 1 è un autovalore semplice per la matrice H''. Inoltre il Teorema 2 implica che l'autovalore dominante di H'' è 1 e l'autovettore associato è l'unico con tutte componenti non negative. Perciò anche l'ultima problematica, ovvero che l'autovalore 1 potrebbe avere più di un autovettore associato, è risolta con quest'ultima modifica di rango uno.

Consideriamo d'ora in poi $A=(H'')^T$.
Le matrici D e E non devono essere esplicitamente calcolate in quanto l'operazione di prodotto matrice-vettore viene eseguita nel seguente modo:

$$\vec{y}=A\vec{x}=(\alpha H'+(1-\alpha)E)^T\vec{x}=$$

$$\alpha H'^T\vec{x}+(1-\alpha)E^T\vec{x}=$$

$$\alpha H^T\vec{x}+\alpha\vec{v}\vec{d}^T\vec{x}+(1-\alpha)\vec{v}\vec{1}^T\vec{x}^T=$$

$$\alpha H^T\vec{x}+w\vec{v}$$

dove $w = (\alpha \vec{d}^T \vec{x} + (1-\alpha)\vec{1}^T \vec{x})$.
Perciò le matrici D ed E non hanno impatto sull'efficienza del codice e sulla sparsità della matrice.

Definizione 7 (Norma-1)**.** Sia $\vec{x} \in \mathbb{C}^n$, $\vec{x} = (x_1, \ldots, x_n)^T$, si definisce la norma-1 di $\vec{x}$ come: $\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$.

Proposizione 1. *Sia A una matrice stocastica per colonna, $\vec{x}$ un vettore tale che $\|\vec{x}\|_1 = 1$. Allora $\|A\vec{x}\|_1 = 1$.*

Osserviamo ora che è possibile riscrivere w come $w = 1 - \|\vec{y}\|_1$, infatti: per la Proposizione 1 si ha che $\|\vec{y}\|_1 = \|A\vec{x}\|_1 = 1$, perciò, se consideriamo la seguente equazione:

$$\vec{y} = \alpha H^T \vec{x} + w\vec{v}$$

e ne calcoliamo la norma-1, moltiplicando per $\vec{1}^T$:

$$1 = \vec{1}^T \alpha H^T \vec{x} + \vec{1}^T w\vec{v},$$

otteniamo:

$$w = 1 - \|\alpha H^T \vec{x}\|_1.$$

L'algoritmo 1 implementa la precedente formula per il calcolo $y = Ax$.

Algoritmo 1

$\vec{y} = \alpha H^T \vec{x}$
$w = 1 - \|\vec{y}\|_1$
$\vec{y} = \vec{y} + w\vec{v}$

1.3 Metodo delle potenze

Il metodo standard per lo sviluppo dell'algoritmo del PageRank è il metodo delle potenze, il cui obiettivo è quello di approssimare l'autovettore dominante della matrice di Hyperlink: l'idea è quella di calcolare esplicitamente $\vec{x}^{(k)} = A\vec{x}^{(k-1)}$ fino a convergenza.
I vantaggi di questo metodo sono molteplici. In particolare, a differenza di altri metodi (come il calcolo dell'inversa o la fattorizzazione QR), non presenta problemi con dimensioni elevate; questa peculiarità risulta essere fondamentale nel caso della matrice di Hyperlink, dove n può raggiungere valori di $O(n^{10})$.

1.3.1 Sviluppo dell'algoritmo

Per semplicità si assume che il vettore $\vec{x}^{(0)}$ appartenga al sottospazio generato dagli autovettori di A. Si scrive, dunque, $\vec{x}^{(0)}$ come combinazione lineare degli autovettori $\vec{v}_1 \dots \vec{v}_n$:

$$\vec{x}^{(0)} = \vec{v}_1 + \alpha_2\vec{v}_2 + ... + \alpha_m\vec{v}_m.$$

Dato che $\vec{x}_0$ è scalato in modo che $\lambda_1 = 1$ si ha:

$$\vec{x}^{(1)} = A\vec{x}^{(0)} = \vec{v}_1 + \alpha_2\lambda_2\vec{v}_2 + ... + \alpha_m\lambda_m\vec{v}_m,$$

$$...$$

$$\vec{x}^{(n)} = A^n\vec{x}^{(0)} = \vec{v}_1 + \alpha_2\lambda_2^n\vec{v}_2 + ... + \alpha_m\lambda_m^n\vec{v}_m.$$

Poiché gli autovalori sono decrescenti ($|\lambda_n| \leq ... \leq |\lambda_2| < 1$) si ha che $A^n\vec{x}^{(0)}$ si avvicina a $\vec{v}_1$ più n è grande. Perciò il metodo delle potenze converge all'autovettore dominante della matrice.

Algoritmo 2: Metodo delle potenze

---

function $\vec{x}^{(n)} = MetodoPotenze()$
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
    $\vec{x}^{(k)} = A\vec{x}^{(k-1)}$;
    $\vec{x}^{(k)} = \vec{x}^{(k)}/\|\vec{x}^{(k)}\|_1$;
    $\delta = \|\vec{x}^{(k)} - \vec{x}^{(k-1)}\|_1$;
    $k = k + 1$;
until $\delta < \epsilon$

1.3.2 Convergenza e costo computazionale

Teorema 4. *Sia $A \in \mathbb{R}^{n\times n}$ e siano λ_1,λ_2,...,λ_n i suoi autovalori ordinati in modo tale che $|\lambda_n| \leq ... \leq |\lambda_2| < |\lambda_1|$, dove λ_1 è semplice. Sia $\vec{x}^{(0)} \in \mathbb{R}^n$ e sia α_1 la componente di $\vec{x}^{(0)}$ rispetto all'autovettore $\vec{v}_1$ associato all'autovalore λ_1. Se $\alpha_1 \neq 0$, allora esiste una costante $C>0$ tale che*

$$\|\vec{x}^{(k)} - \vec{v}_1\|_2 \leq C\frac{|\lambda_2|}{|\lambda_1|}^k, k \geq 1.$$

Il teorema appena enunciato fornisce una stima della velocità di convergenza della successione $\left\{\vec{x}^{(k)}\right\}$ all'autovettore $\vec{v}_1$. Mostra inoltre che tale convergenza è lineare rispetto a $\dfrac{|\lambda_2|}{|\lambda_1|}$.

La convergenza, quindi, dipende da $\dfrac{|\lambda_2|}{|\lambda_1|} = |\lambda_2| \leq \alpha$ con $\alpha \in (0, 1)$, dove α è il parametro usato nel Teorema 3. Tenendo presente che valori di α lontani da 1 garantiscono una convergenza più veloce e che per valori di α vicini a 1 si ha maggior attinenza alla struttura del Web, è possibile intuire quanto sia fondamentale determinare il giusto valore di α per ottenere dei risultati ottimali.

Analizzando l'algoritmo 1, sviluppato per il calcolo di $A\vec{x}$, si può notare che le matrici H'' e H' artificialmente introdotte non vengono effettivamente formate; perciò si ha una riduzione del costo computazionale in quanto creare la matrice H'' e utilizzarla per il prodotto matrice-vettore sarebbe molto più dispendioso. Grazie alla sparsità della matrice H, inoltre, il costo computazionale si riduce ulteriormente: da $(O(n^2))$ passa a $O(n)$.

1.3.3 Risultati sperimentali

Il dataset utilizzato è Stanford-Berkeley, si tratta di una matrice di dimensioni 683446×683446 contenente circa 7.6 milioni di links.
Come si può notare dalla *Figura 1* la matrice è sparsa; questo permette di ridurre notevolmente il tempo di calcolo e il costo computazionale.

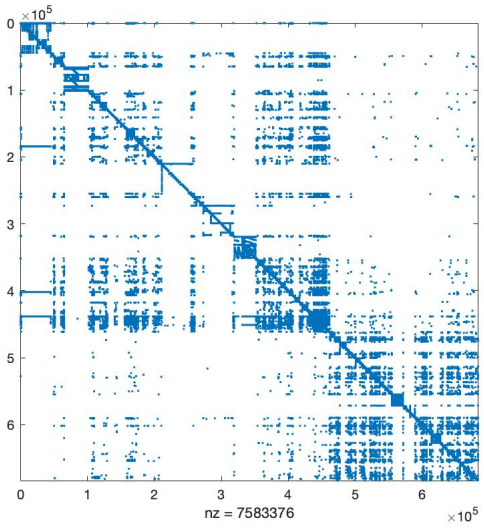


Figura 1: Grafico degli elementi non nulli della matrice

Analizziamo ora il metodo delle potenze con diversi valori di α,
$\alpha \in \{0.50, 0.85, 0.95, 0.99\}$.
I risultati mostrati nella *Figura 2* confermano quanto visto nella sezione precedente: si ha una convergenza più veloce per i valori di α più lontani da 1.

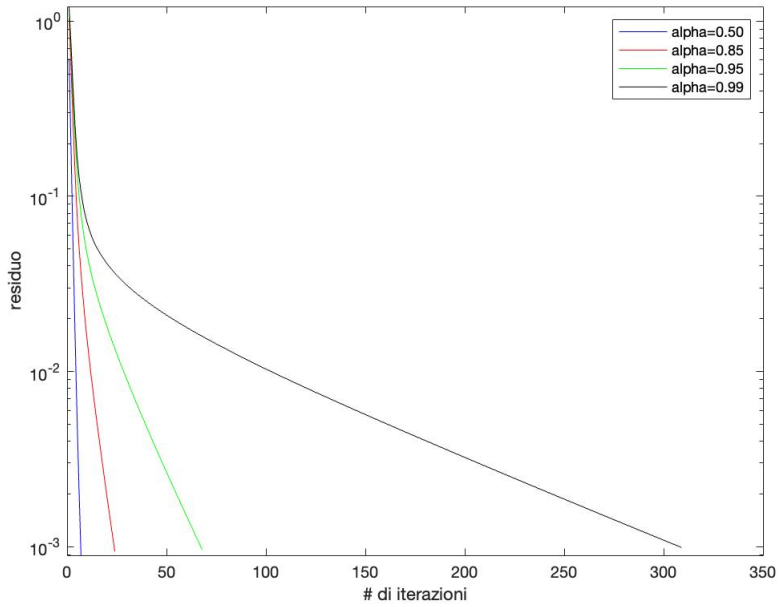


Figura 2: Metodo delle potenze con diversi α

Inoltre anche per quanto riguarda i tempi di calcolo si ha una notevole differenza fra il valore estremo $\alpha = 0.99$ ed altri valori più vicini a 0, come mostrato nella *Figura 3*.

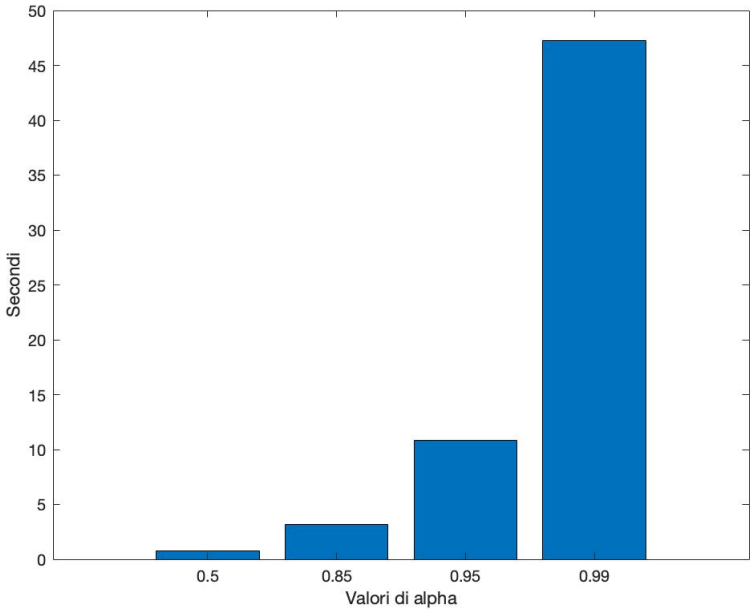


Figura 3: Tempi calcolo del metodo delle potenze con diversi α

Ricordiamo che il vantaggio di utilizzare valori di α più grandi è quello di avere più attinenza alla struttura del web, di conseguenza la scelta di $\alpha = 0.50$ non è ottimale poiché si discosta dalla realtà. Possiamo concludere quindi che un giusto compromesso sia $\alpha = 0.85$.
Il metodo delle potenze assicura la convergenza. Tuttavia, data la vastità del Web, usare tale metodo potrebbe richiedere diversi giorni di calcolo mentre l'utente richiede una risposta in millisecondi.
Nel prossimo capitolo vengono presentati tre metodi che migliorano l'efficacia di ogni iterazione del metodo delle potenze, rimuovendo l'errore della componente di $\vec{x}^{(k)}$ lungo le direzioni $\vec{v}_2$ e $\vec{v}_3$.

Capitolo 2

Estrapolazione

I metodi di estrapolazione si basano sull'esprimere l'iterata $\vec{x}^{(k)}$ come combinazione lineare del tipo $\vec{x}^{(k)} = \vec{v}_1 + \lambda_2\vec{v}_2 + \cdots + \lambda_m\vec{v}_m + \ldots$, dove i $\vec{v}_i$ sono gli autovettori e i λ_i sono gli autovalori della matrice A.
Kamvar presentò diversi metodi di estrapolazione, la cui differenza principale risiede nel numero di autovettori con cui si approssima l'iterata; nel caso dell'estrapolazione di Aitken e dell'estrapolazione epsilon l'iterata $\vec{x}^{(k-2)}$ viene approssimata come combinazione lineare dei primi due autovettori, nell'estrapolazione quadratica dei primi tre.
I metodi sono applicati periodicamente per accelerare la convergenza del metodo delle potenze.

2.1 Estrapolazione di Aitken

2.1.1 Sviluppo dell'algoritmo

Come detto in precedenza l'estrapolazione di Aitken si basa sull'idea di poter calcolare l'autovettore dominante della matrice supponendo di poter esprimere l'iterata $\vec{x}^{(k-2)}$ come combinazione lineare dei primi due autovettori. Sotto tale ipotesi è quindi possibile calcolare l'autovettore dominante $\vec{v}_1$, usando le iterate successive $\vec{x}^{(k-2)} \ldots \vec{x}^{(k)}$.
Si ha perciò:

$$\vec{x}^{(k-2)} = \vec{v}_1 + \alpha_2\vec{v}_2. \tag{2.1}$$

Dato che il primo autovalore, $\lambda_1 = 1$, le iterate successive possono essere espresse come:

$$\vec{x}^{(k-1)} = A\vec{x}^{(k-2)} = \vec{v}_1 + \alpha_2\lambda_2\vec{v}_2, \tag{2.2}$$

$$\vec{x}^{(k)} = A\vec{x}^{(k-1)} = \vec{v}_1 + \alpha_2\lambda_2^2\vec{v}_2. \tag{2.3}$$

Si definisca ora:

$$g_i := (x_i^{(k-1)} - x_i^{(k-2)})^2, \tag{2.4}$$

$$h_i := x_i^{(k)} - 2x_i^{(k-1)} + x_i^{(k-2)}. \tag{2.5}$$

Utilizzando le equazioni (2.2) e (2.3) è possibile ridefinire le equazioni (2.4) e (2.5) come segue:

$$g_i = \alpha_2^2(\lambda_2 - 1)^2(v_2)_i^2$$

$$h_i = \alpha_2(\lambda_2 - 1)^2(v_2)_i.$$

Si definisca ora f_i come il quoziente fra g_i e h_i

$$f_i := \frac{g_i}{h_i} = \frac{\alpha_2^2(\lambda_2 - 1)^2(v_2)_i^2}{\alpha_2(\lambda_2 - 1)^2(v_2)_i} = \alpha_2(v_2)_i,$$

perciò

$$\vec{f} = \alpha_2\vec{v}_2.$$

Infine, dall'equazione (2.1), si ottiene un'approssimazione di $\vec{v}_1$:

$$\vec{v}_1 = \vec{x}^{(k-2)} - \alpha_2\vec{v}_2 = \vec{x}^{(k-2)} - \vec{f}.$$

Gli algoritmi 3 e 4 mostrano come combinare l'estrapolazione di Aitken con il metodo delle potenze.
L'algoritmo 3 calcola l'iterata successiva mediante l'estrapolazione di Aitken, sfruttando le definizioni di g_i e h_i, rispettivamente in (2.4) e (2.5).

Algoritmo 3: calcolo iterata successiva

---

function $\vec{x}^*$=Aitken($\vec{x}^{(k-2)}$,$\vec{x}^{(k-1)}$,$\vec{x}^{(k)}$)
for i=1:n **do**
  $g_i = (x_i^{(k-1)} - x_i^{(k-2)})^2$;
  $h_i = x_i^{(k)} - 2x_i^{(k-1)} + x_i^{(k-2)}$;
  $x_i = x_i^{(k)} - g_i/h_i$;
end

L'algoritmo 4 mostra, invece, come applicare periodicamente l'estrapolazione di Aitken al metodo delle potenze.

Algoritmo 4: Metodo potenze con estrapolazione di Aitken

function $\vec{x}^{(n)}$=AitkenMetodoPotenze()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}^{(k)} = A\vec{x}^{(k-1)}$;
  $\vec{x}^{(k)} = \vec{x}^{(k)} / \|\vec{x}^{(k)}\|_1$;
  $\delta = \|\vec{x}^{(k)} - \vec{x}^{(k-1)}\|_1$;
  periodically, $\vec{x}^{(k)}$=Aitken($\vec{x}^{(k-2)}$,$\vec{x}^{(k-1)}$,$\vec{x}^{(k)}$);
  $k = k + 1$;
until $\delta < \varepsilon$

2.1.2 Costo computazionale

Affinché il metodo sia ottimale ogni costo aggiuntivo (overhead) al costo dell'algoritmo 1 deve essere minimale. Appare evidente che il costo di un ciclo dell'algoritmo 3 è $O(n)$. Il costo di ogni iterata dell'estrapolazione è, quindi, minore di una singola iterazione del metodo delle potenze; inoltre, poiché l'estrapolazione viene applicata solo in modo periodico la condizione sopra citata è soddisfatta.

2.1.3 Risultati sperimentali

Si vuole ora analizzare la differenza fra il metodo delle potenze e l'estrapolazione di Aitken, applicata alla decima iterata, con $\alpha = 0.99$.
Come si può notare dalla *Figura 4*, la convergenza avviene più velocemente con l'estrapolazione di Aitken. Si osserva, infatti, che dopo un picco iniziale del residuo, dovuto al fatto che $\vec{v}_2$ è approssimato in maniera poco precisa, il metodo accelerato è più efficiente.

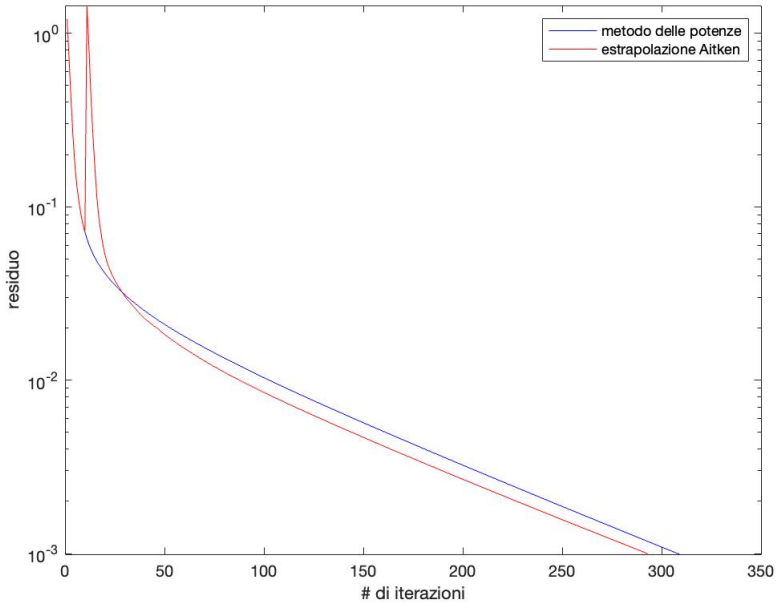


Figura 4: Confronto metodo potenze e estrapolazione di Aitken con $\alpha = 0.99$

Per α minori, invece, l'estrapolazione perde ogni vantaggio infatti, come mostrato nella *Figura 5* in cui $\alpha = 0.80$, si ha che il metodo delle potenze converge prima. Inoltre l'estrapolazione perde vantaggio anche quando viene applicata troppe volte, come espresso in [1], questo è dovuto ai picchi del residuo che impediscono una convergenza veloce. Questo è evidente nella *Figura 6*, dove applicando l'estrapolazione ogni dieci iterazioni non si giunge a convergenza.

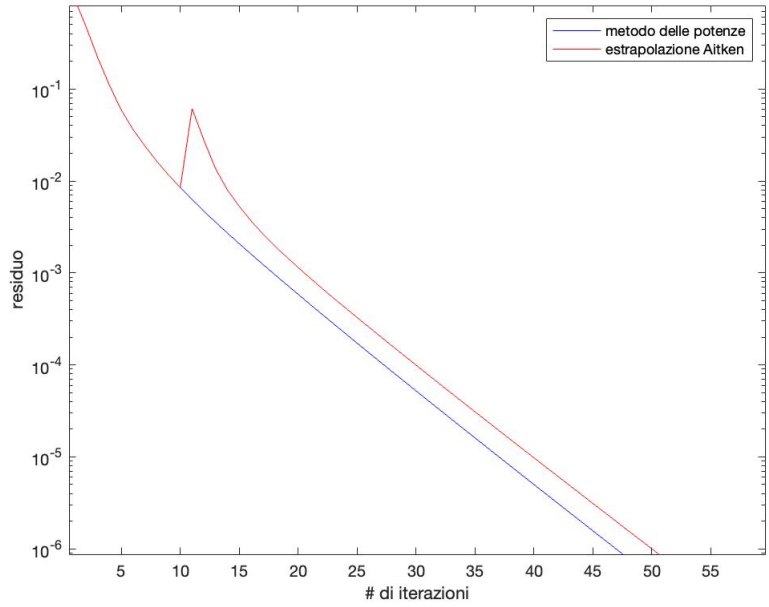


Figura 5: Confronto metodo potenze e estrapolazione di Aitken con $\alpha = 0.80$

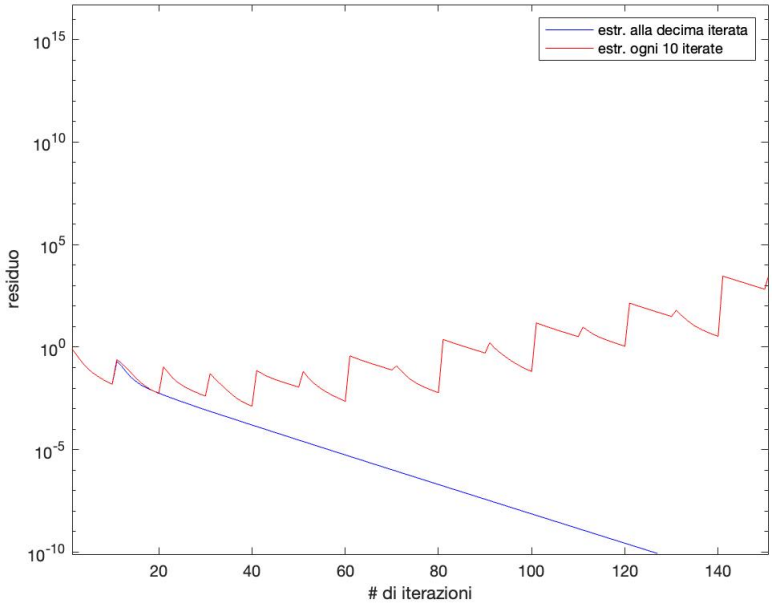


Figura 6: Confronto fra l'applicazione dell'estrapolazione di Aitken alla decima iterata e ogni dieci iterate

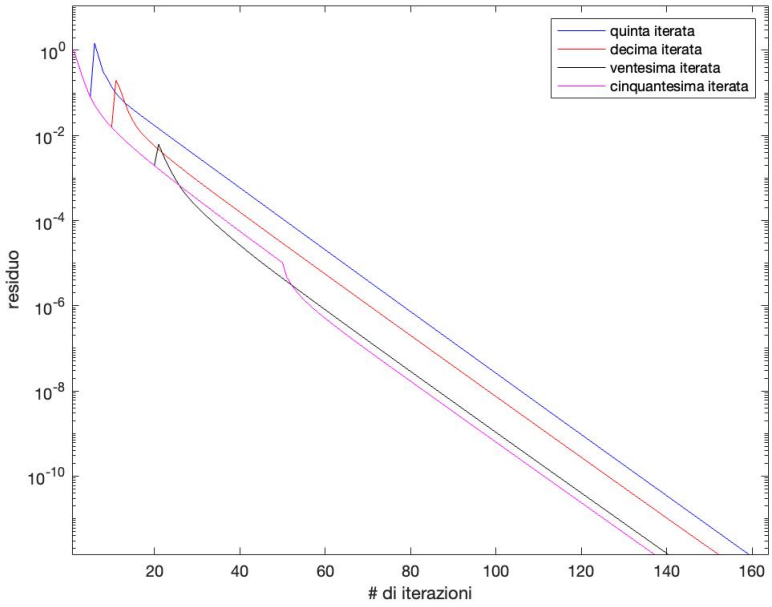


Figura 7: Confronto applicazione estrapolazione a diverse iterate con $\alpha = 0.85$

Il grafico in *Figura 7* studia il diverso comportamento del residuo del metodo di Aitken applicato in diverse iterazioni. Lo scopo è quello di cercare il numero di iterazioni ottimale per arrivare a convergenza. Come si può notare, se $\alpha = 0.85$, si ha più vantaggio ad applicare l'estrapolazione ad iterate più grandi: questo è dovuto al fatto che il picco iniziale rallenta la convergenza del metodo.

2.2 Estrapolazione epsilon

Questa estrapolazione è molto simile all'estrapolazione di Aitken, infatti, anche in questo caso, si va ad approssimare l'iterata $\vec{x}^{(k-2)}$ come combinazione lineare dei primi due autovettori della matrice A: la differenza principale risiede nella definizione di g_i.

2.2.1 Sviluppo algoritmo

Si procede come per l'estrapolazione di Aitken, andando quindi ad supporre

$$\vec{x}^{(k-2)} = \vec{v}_1 + \alpha_2 \vec{v}_2,$$

scrivendo le iterate successive:

$$\vec{x}^{(k-1)} = A\vec{x}^{(k-2)} = \vec{v}_1 + \alpha_2 \lambda_2 \vec{v}_2,$$

$$\vec{x}^{(k)} = A\vec{x}^{(k-1)} = \vec{v}_1 + \alpha_2\lambda_2^2\vec{v}_2.$$

Si definisca ora:

$$g_i := (x_i^{(k-1)} - x_i^{(k-2)})(x_i^{(k)} - x_i^{(k-1)}) = \alpha_2^2\lambda_2(\lambda_2 - 1)^2(v_2)_i^2.$$

Mentre h_i è analogo alla definizione data nella sezione precedente:

$$h_i = \alpha_2(\lambda_2 - 1)^2(v_2)_i.$$

Da cui si ha:

$$f_i = \frac{g_i}{h_i} = \frac{\alpha_2^2\lambda_2(\lambda_2 - 1)^2(v_2)_i^2}{\alpha_2(\lambda_2 - 1)^2(v_2)_i} = \alpha_2\lambda_2(v_2)_i$$

ottenendo quindi l'approssimazione di $\vec{v}_1$:

$$\vec{v}_1 = \vec{x}^{(k-1)} - \alpha_2\lambda_2\vec{v}_2 = \vec{x}^{(k-1)} - \vec{f}.$$

Sono qui di seguito riportati i due algoritmi che permettono di risolvere il problema: l'algoritmo 5 calcola l'iterata successiva utilizzando l'estrapolazione epsilon, l'algoritmo 6 risolve utilizzando il metodo delle potenze in cui viene calcolata l'iterata, periodicamente, con l'estrapolazione.

Algoritmo 5: calcolo iterata successiva

---

function $\vec{x}^*$=Epsilon($\vec{x}^{(k-2)}$,$\vec{x}^{(k-1)}$,$\vec{x}^{(k)}$)
for i=1:n **do**
  $g_i = (x_i^{(k-1)} - x_i^{(k-2)})(x_i^{(k)} - x_i^{(k-1)})$;
  $h_i = x_i^{(k)} - 2x_i^{(k-1)} + x_i^{(k-2)}$;
  $x_i = x_i^{(k)} - g_i/h_i$;
end

Algoritmo 6: Metodo potenze con estrapolazione epsilon

function $\vec{x}^{(n)}$=EpsilonMetodoPotenze()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}^{(k)} = A\vec{x}^{(k-1)}$;
  $\vec{x}^{(k)} = \vec{x}^{(k)} / \|\vec{x}^{(k)}\|_1$;
  $\delta = \|\vec{x}^{(k)} - \vec{x}^{(k-1)}\|_1$;
  periodically, $\vec{x}^{(k)}$=Epsilon($\vec{x}^{(k-2)}$,$\vec{x}^{(k-1)}$,$\vec{x}^{(k)}$);
  $k = k + 1$;
until $\delta < \varepsilon$

2.2.2 Costo computazionale

Abbiamo quindi osservato che l'unica differenza rispetto al metodo delle potenze calcolato con l'estrapolazione di Aitken risiede nel calcolo di g_i, il quale ha un costo di $3n$. Si ha perció che il costo di un'iterazione è $O(n)$ ottenendo anche in questo caso, un overhead minimo.

2.2.3 Risultati sperimentali

Nonostante la differenza con il metodo di Aitken sia solo nella definizione di g_i si ha, con il dataset "Stanford-Berkeley", una differenza notevole per quanto riguarda la convergenza.
Come si può osservare nella *Figura 8*, infatti, l'estrapolazione epsilon con $\alpha = 0.85$ applicata alla quarantesima iterata converge addirittura sia dopo il metodo delle potenze sia dopo il metodo con estrapolazione di Aitken.

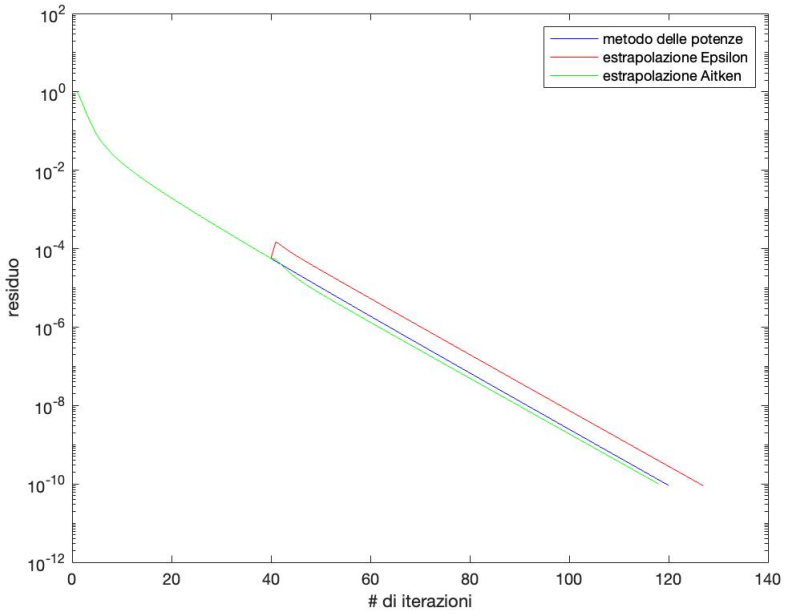


Figura 8: Confronto metodo potenze, estrapolazione Aitken ed epsilon

Per quanto riguarda la periodicità dell'estrapolazione il metodo epsilon si comporta come Aitken: vi è sempre presente, infatti, il picco iniziale del residuo che rallenta la convergenza del metodo, per questo l'applicare troppo spesso o troppo presto l'estrapolazione non porta a risultati soddisfacenti.

La *Figura 9* riporta un confronto fra l'estrapolazione applicata alla decima iterata ed applicata ogni dieci iterate. Si osserva dunque che, in questo caso, il numero di iterazioni per la convergenza viene più che raddoppiato.
Infine nella *Figura 10* è messo in evidenza come la convergenza sia più veloce se si applica l'estrapolazione ad iterate maggiori.

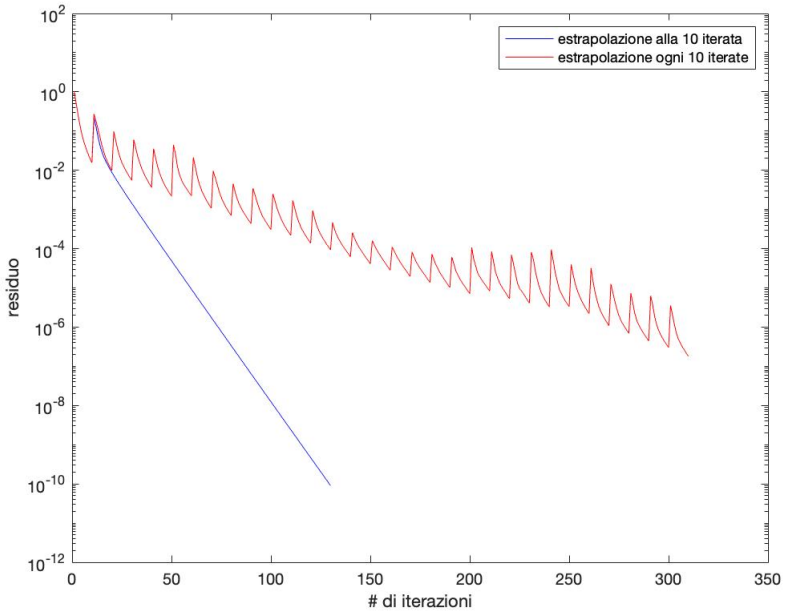


Figura 9: Confronto estrapolazione epsilon applicata alla decima iterata e ogni dieci iterate

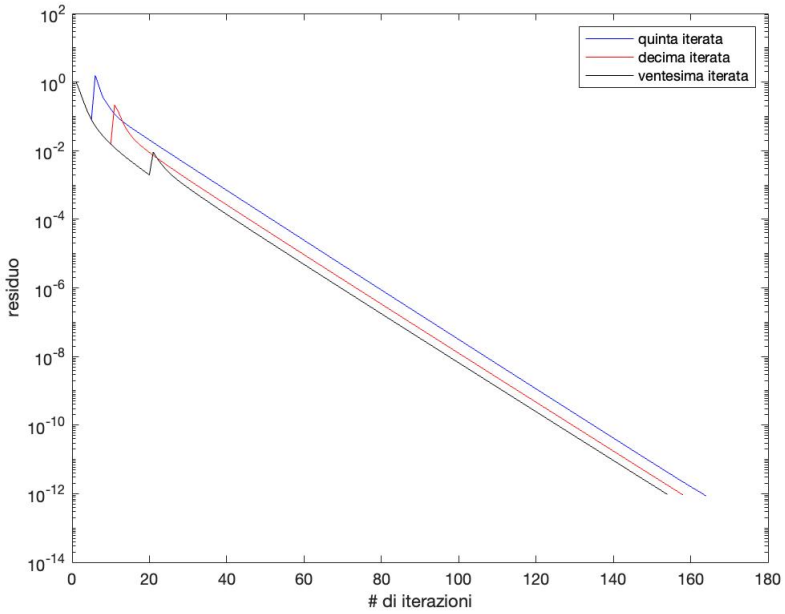


Figura 10: Confronto estrapolazione epsilon con diverse iterate

Nella prossima sezione viene presentata l'estrapolazione quadratica. L'idea principale di questo metodo è quella di approssimare l'iterata come combinazione lineare dei primi tre autovettori.

2.3 Estrapolazione quadratica

Si supponga ora di poter approssimare l'iterata $\vec{x}^{(k-3)}$ come combinazione lineare di tre autovettori; sotto questa ipotesi è possibile andare a stimare il valore dell'autovettore dominante usando le iterate succesive $\vec{x}^{(k-3)} \dots \vec{x}^{(k)}$. Approssimando con tre autovettori e non con due, come si può osservare nei risultati sperimentali riportati in seguito, non si ha più il problema del picco iniziale del residuo, si ottiene quindi un metodo che converge più velocemente rispetto all'estrapolazione di Aitken.

2.3.1 Sviluppo dell'algoritmo

Assumiamo, quindi, che la matrice A abbia solo tre autovettori, e scriviamo l'iterata $\vec{x}^{(k-3)}$ come combinazione lineare di essi:

$$\vec{x}^{(k-3)} = \vec{v}_1 + \alpha_2\vec{v}_2 + \alpha_3\vec{v}_3.$$

Definiamo ora le iterate successive:

$$\vec{x}^{(k-2)} = A\vec{x}^{(k-3)}; \vec{x}^{(k-1)} = A\vec{x}^{(k-2)}; \vec{x}^{(k)} = A\vec{x}^{(k-1)}. \tag{2.6}$$

Dato che la matrice ha solo tre autovettori, il polinomio caratteristico ha grado 3 ed è il seguente:

$$p_A(\lambda) = \gamma_0 + \gamma_1\lambda + \gamma_2\lambda^2 + \gamma_3\lambda^3.$$

La matrice A ha come autovalore $\lambda_1 = 1$, perciò

$$p_A(1) = 0 \Rightarrow \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3 = 0. \tag{2.7}$$

Utilizziamo ora il seguente teorema:

Teorema 5 (Cayley-Hamilton)**.** *Sia A, una matrice quadrata, $A \in \mathbb{K}^{n,n}$, e $p_A(x)$ il suo polinomio caratteristico, allora $p_A(A) = 0$.*

Dal Teorema 5 segue che, preso un qualsiasi vettore $\vec{z} \in \mathbb{R}^n$ vale $p_A(A)\vec{z} = 0$. Ponendo $\vec{z} = \vec{x}^{(k-3)}$ si ottiene:

$$p_A(A)\vec{x}^{(k-3)} = 0 \Rightarrow [\gamma_0 I + \gamma_1 A + \gamma_2 A^2 + \gamma_3 A^3]\vec{x}^{(k-3)} = 0.$$

Sfruttando le definizioni delle iterate successive (2.6):

$$\gamma_0\vec{x}^{(k-3)} + \gamma_1\vec{x}^{(k-2)} + \gamma_2\vec{x}^{(k-1)} + \gamma_3\vec{x}^{(k)} = 0.$$

Dall'equazione (2.7) si ha:

$$(\vec{x}^{(k-2)} - \vec{x}^{(k-3)})\gamma_1 + (\vec{x}^{(k-1)} - \vec{x}^{(k-3)})\gamma_2 + (\vec{x}^{(k)} - \vec{x}^{(k-3)})\gamma_3 = 0. \tag{2.8}$$

Definendo $\vec{y}^{(k-2)} = \vec{x}^{(k-2)} - \vec{x}^{(k-3)}$; $\vec{y}^{(k-1)} = \vec{x}^{(k-1)} - \vec{x}^{(k-3)}$; $\vec{y}^{(k)} = \vec{x}^{(k)} - \vec{x}^{(k-3)}$; è possibile ridefinire (2.8) in notazione matriciale:

$$(\vec{y}^{(k-2)} \quad \vec{y}^{(k-1)} \quad \vec{y}^{(k)})\ \vec{\gamma} = 0. \tag{2.9}$$

Siamo interessati a risolvere questa equazione per $\vec{\gamma} \neq 0$.
Si osservi che ponendo $\gamma_3 = 1$ (ipotesi non restrittiva poiché fissare un singolo termine del polinomio caratteristico non influenza gli zeri) l'equazione (2.9) diventa:

$$(\vec{y}^{(k-2)} \quad \vec{y}^{(k-1)})\ \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = -\vec{y}^{(k)}.$$

Si tratta di un sistema sovradeterminato, quindi risolviamo il corrispondente problema ai minimi quadrati:

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = Y^+\vec{y}^{(k)}$$

dove Y^+ è la matrice pseudoinversa di $Y = (\vec{y}^{(k-2)} \quad \vec{y}^{(k-1)})$:

Definizione 8 (Matrice pseudoinversa)**.** Sia $A \in \mathbb{R}^{m\times n}$, la pseudoinversa di A è l'unica matrice $n \times m$, donotata con A^+, che verifica le seguenti proprietà:

- $AA^+A = A$
- $A^+AA^+ = A^+$
- $(AA^+)^T = AA^+$
- $(A^+A)^T = A^+A$.

Si divida ora il polinomio caratteristico $p_A(\lambda)$ per $\lambda - 1$

$$q_A(\lambda) = \frac{p_A(\lambda)}{\lambda - 1} = \frac{\gamma_0 + \gamma_1\lambda + \gamma_1\lambda^2 + \gamma_1\lambda^3}{\lambda - 1} = \beta_0 + \beta_1\lambda + \beta_2\lambda^2$$

dove:

$$\beta_0 = \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3; \beta_1 = \gamma_2 + \gamma_3; \beta_2 = \gamma_3.$$

Dal teorema di Cayley-Hamilton si ha quindi che dato un qualsiasi $\vec{z} \in \mathbb{R}^n$, $q_A(A)\vec{z} = \vec{v}_1$; perciò ponendo $\vec{z} = \vec{x}^{(k-2)}$:

$$\vec{v}_1 = q_A(A)\vec{x}^{(k-2)} = [\beta_0 I + \beta_1 A + \beta_2 A^2]\vec{x}^{(k-2)}.$$

Infine, dalle equazioni (2.6), otteniamo l'approssimazione:

$$\vec{u}_1 = \beta_0\vec{x}^{(k-2)} + \beta_1\vec{x}^{(k-1)} + \beta_2\vec{x}^{(k)}.$$

L'algoritmo 7 calcola l'iterata successiva utilizzando la teoria appena enunciata; tale algoritmo viene utilizzato poi nell'algoritmo 8 che combina il metodo delle potenze con l'estrapolazione quadratica, applicata periodicamente.

Algoritmo 7: calcolo iterata successiva

$\vec{x}^*$=QuadraticExtrapolation($\vec{x}^{(k-3)}$,$\vec{x}^{(k-2)}$,$\vec{x}^{(k-1)}$,$\vec{x}^{(k)}$)
for j=k-2:k **do**
  $\vec{y}^{(j)} = \vec{x}^{(j)} - \vec{x}^{(k-3)}$;
end
$Y = \begin{pmatrix} \vec{y}^{(k-2)} & \vec{y}^{(k-1)} \end{pmatrix}$;
$\gamma_3 = 1$;
$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = -Y^+\vec{y}^{(k)}$;
$\gamma_0 = -(\gamma_1 + \gamma_2 + \gamma_3)$;
$\beta_0 = \gamma_1 + \gamma_2 + \gamma_3$;
$\beta_1 = \gamma_2 + \gamma_3$;
$\beta_3 = \gamma_3$;
$\vec{x}^* = \beta_0\vec{x}^{(k-2)} + \beta_1\vec{x}^{(k-1)} + \beta_2\vec{x}^{(k)}$;
end

Algoritmo 8: Metodo potenze con estrapolazione quadratica

function $\vec{x}^n$=QuadraticMetodoPotenze()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}^{(k)} = A\vec{x}^{(k-1)}$;
  $\vec{x}^{(k)} = \vec{x}^{(k)} / \|\vec{x}^{(k)}\|_1$;
  $\delta = \|\vec{x}^{(k)} - \vec{x}^{(k-1)}\|_1$;
  periodically, $\vec{x}^{(k)}$=QuadraticExtrapolation($\vec{x}^{(k-3)}$,...,$\vec{x}^{(k)}$);
  $k = k + 1$;
until $\delta < \varepsilon$;

2.3.2 Costo computazionale

Analizzando l'algoritmo 6 si osserva che tutti i calcoli sono $O(1)$ o $O(n)$ ad esclusione del problema ai minimi quadrati:

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = Y^+ \vec{y}^{(k)}$$

che è quindi, in termini di costo computazionale, l'operazione che provoca un overhead maggiore rispetto al metodo delle potenze standard.
Tuttavia, dato che Y è una matrice $n \times 2$, è possibile ridurre il costo di tale calcolo utilizzando l'algoritmo di Gram-Schmidt come segue:

1. applicare Gram-Schmidt per ottenere la fattorizzazione QR della matrice Y: $Y = QR$,
2. calcolare il vettore $-Q^T \vec{y}^{(k)}$,
3. risolvere il sistema triangolare superiore usando la sostituzione all'indietro: $R \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = -Q^T \vec{y}^{(k)}$.

In questo modo anche il problema ai minimi quadrati ha un costo computazionale di $O(n)$.
Dato che l'estrapolazione è applicata in modo periodico ed ogni operazione aggiuntiva è dell'ordine di una singola iterazione del metodo delle potenze, si ha un overhead minimo.

2.3.3 Risultati sperimentali

L'estrapolazione quadratica migliora i risultati del metodo delle potenze standard quanto più α è vicino ad 1. Questo risultato è possibile notarlo nelle *Figure 11 e 12*, dove è presentato un confronto fra il metodo delle potenze e l'estrapolazione quadratica con due α diversi nei due grafici. Essendo quindi, nella *Figura 11* α più vicino ad 1, si ha una convergenza molto più veloce rispetto alla *Figura 12*: in quest'ultima si può notare come il metodo delle potenze converga addirittura più velocemente. L'estrapolazione, quindi, per α lontani da 1 perde il suo vantaggio. In entrambe le figure l'estrapolazione è applicata le prime 10 volte possibili.

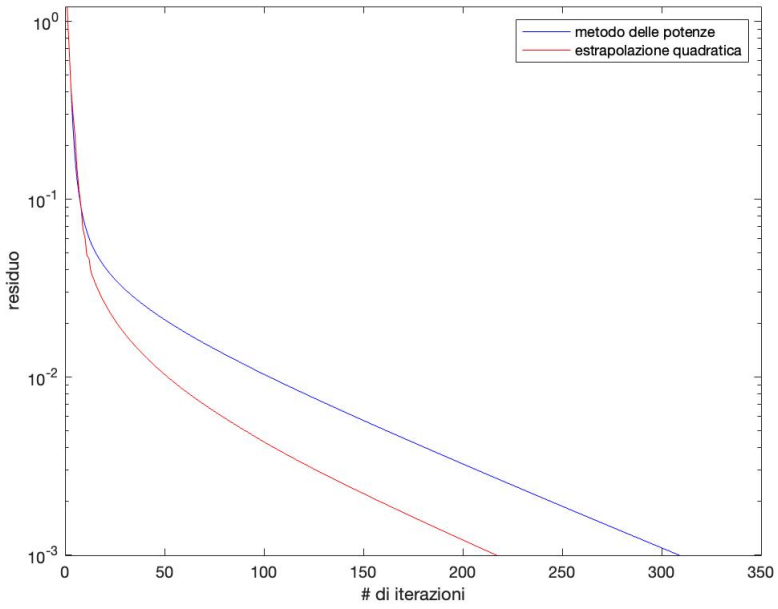


Figura 11: Confronto metodo potenze e estrapolazione quadratica con $\alpha = 0.99$

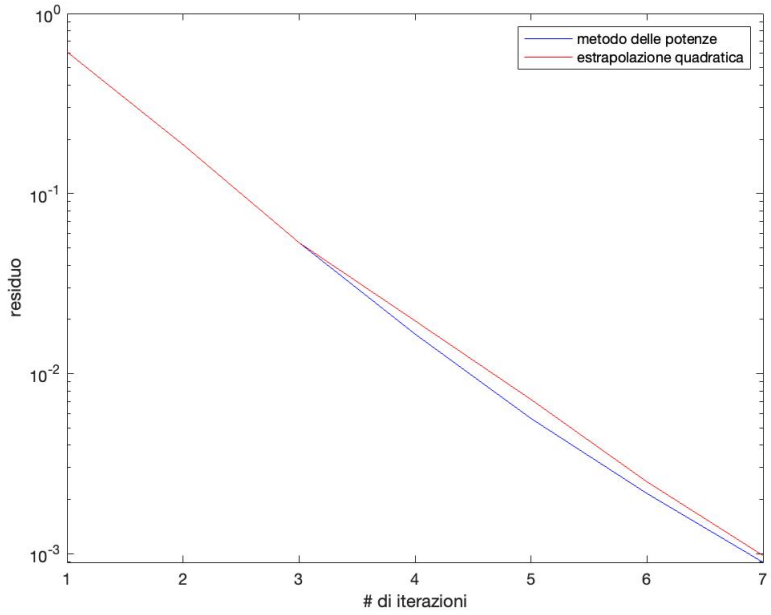


Figura 12: Confronto metodo potenze e estrapolazione quadratica con $\alpha = 0.50$

Come è possibile notare dalla *Figura 13* non è necessario applicare spesso l'estrapolazione per ottenere dei buoni risultati, anzi questo aumenta sia il costo computazionale sia il tempo di calcolo. Come si può osservare, con il dataset "Stanford-Berkeley" e con $\alpha = 0.85$, si ha convergenza più veloce se si applica l'estrapolazione ogni 40 iterazioni.

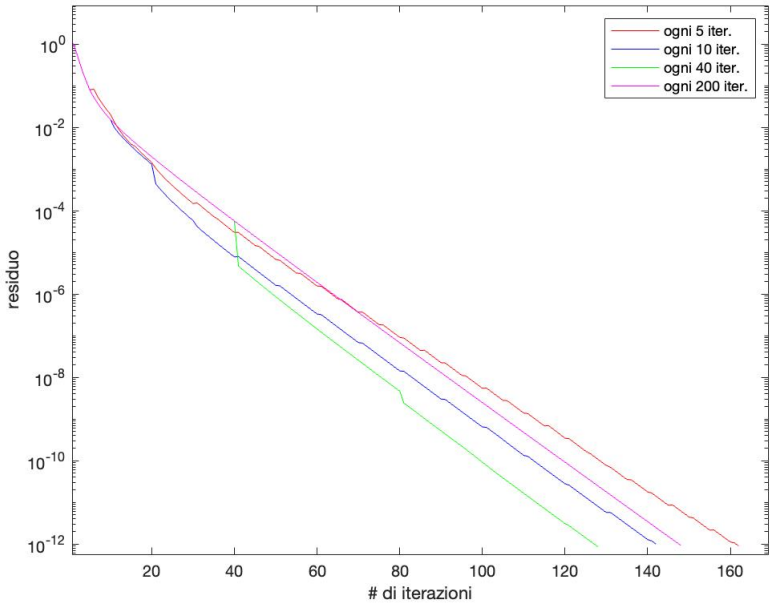


Figura 13: Estrapolazione quadratica applicata con diverse periodicità

Capitolo 3

Metodi adattivi

Diciamo che il PageRank x_i di una pagina i converge quando

$$\frac{|x_i^{(k+1)} - x_i^{(k)}|}{|x_i^{(k)}|} < \varepsilon$$

dove ε è una tolleranza fissata.
Kamvar, Haveliwala e Golub notarono che non tutte le pagine web convergono al loro rispettivo PageRank nello stesso momento. È, quindi, possibile migliorare ulteriormente l'algoritmo del PageRank, considerando che:

- non si ha la necessità di ricalcolare il contributo del PageRank di pagine che sono già giunte a convergenza,
- se si è interessati a calcolare il PageRank di una pagina non è necessario ricalcolare il contributo del PageRank di pagine ad essa collegate già giunte a convergenza.

In questo capitolo si introduce l'algoritmo adattivo che modifica il calcolo del PageRank precedente usando queste due osservazioni.

3.1 Sviluppo dell'algoritmo

Supponiamo di aver completato la k-esima iterazione del metodo delle potenze, usando l'iterata $\vec{x}^{(k)}$ vogliamo calcolare $\vec{x}^{(k+1)}$.
Sia C l'insieme delle pagine che sono giunte a convergenza data una certa tolleranza; sia N l'insieme delle pagine che non sono ancora giunte a convergenza.
È possibile suddividere la matrice A in due sottomatrici:

- A_N: matrice $m \times n$ corrispondente agli inliks (links entranti) delle m pagine il cui pagerank non è ancora giunto a convergenza,
- A_C: matrice $(n-m) \times n$ corrispondente agli inliks delle pagine il cui pagerank è giunto a convergenza.

Si può fare un discorso analogo con il vettore di PageRank $\vec{x}^{(k)}$:

- $\vec{x}_N^{(k)}$: vettore di lunghezza m corrispondente alle componenti di $\vec{x}^{(k)}$ che non sono ancora giunte a convergenza,
- $\vec{x}_C^{(k)}$: vettore di lunghezza $(n-m)$ corrispondente alle componenti di $\vec{x}^{(k)}$ che sono giunte a convergenza.

Perciò riordiniamo A e $\vec{x}^{(k)}$:

$$A = \begin{pmatrix} A_N \\ A_C \end{pmatrix},$$

$$\vec{x}^{(k)} = \begin{pmatrix} \vec{x}_N^{(k)} \\ \vec{x}_C^{(k)} \end{pmatrix}.$$

Quindi possiamo riscrivere l'iterata del metodo delle potenze come:

$$\begin{pmatrix} \vec{x}_N^{(k+1)} \\ \vec{x}_C^{(k+1)} \end{pmatrix} = \begin{pmatrix} A_N \\ A_C \end{pmatrix} \begin{pmatrix} \vec{x}_N^{(k)} \\ \vec{x}_C^{(k)} \end{pmatrix}.$$

Dato che gli elementi di $\vec{x}_C^{(k)}$ sono già giunti a convergenza, non è necessario calcolare $\vec{x}_C^{(k+1)}$, perciò:

$$\vec{x}_N^{(k+1)} = A_N \vec{x}^{(k)},$$
$$\vec{x}_C^{(k+1)} = \vec{x}_C^{(k)}.$$

Algoritmo 9: Metodo Adattivo PageRank

function $\vec{x}^{(n)}$=MetodoAdattivo()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}_N^{(k+1)} = A_N \vec{x}^{(k)}$;
  $\vec{x}_C^{(k+1)} = \vec{x}_C^{(k)}$;
  $[N, C]$ =SelezionaConvergenti($\vec{x}^{(k)}, \vec{x}^{(k+1)}, \varepsilon$);
  periodically $\delta = \|A\vec{x}^{(k)} - \vec{x}^{(k)}\|_1$;
  $k = k + 1$;
until $\delta < \varepsilon$

3.1.1 Costo computazionale

Identificare ad ogni iterazione le pagine che sono giunte a convergenza non è costoso, tuttavia è molto costoso riordinare la matrice A ad ogni iterazione. Dato che A_N è più piccola di A il costo della moltiplicazione per il vettore è ridotto, quindi è auspicabile moltiplicare solo per A_N e non per tutta A.

3.2 Metodo adattivo filtrato

Come detto in precedenza creare la sottomatrice A_N ad ogni iterazione è troppo costoso perciò descriviamo ora un algoritmo efficiente per l'implementazione del metodo adattivo.
Considerando le notazioni della sezione precedente, si può osservare che la sottomatrice A_C non viene mai utilizzata per il calcolo di $\vec{x}^{(k+1)}$. Definiamo, quindi, la matrice A' in cui sostituisco ad A_C una matrice di 0 delle stesse dimensioni:

$$A' = \begin{pmatrix} A_N \\ 0 \end{pmatrix}.$$

Analogamente definiamo:

$$\vec{x}_C^{\prime(k)} = \begin{pmatrix} \vec{0} \\ \vec{x}_C^{(k)} \end{pmatrix}.$$

Perció possiamo calcolare l'iterata $\vec{x}^{(k+1)}$ come segue:

$$\vec{x}^{(k+1)} = A' \vec{x}^{(k)} + \vec{x}_C^{\prime(k)}.$$

L'idea che sta alla base del nuovo algoritmo è che aumentando la sparsità della matrice A possiamo diminuire sensibilmente il costo computazionale.
Sia C un insieme di indici, corrispondenti alle pagine che sono state identificate come convergenti.
Definiamo la matrice A'':

$$A''_{ij} = \begin{cases} 0 & \text{se i} \in \text{C}, \\ A_{ij} & \text{altrimenti.} \end{cases}$$

Analogamente, definiamo $\vec{x}_C^{\,\prime\prime(k)}$:

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

Perció, considerando queste nuove definizioni, l'iterazione diventa:

$$\vec{x}^{(k+1)} = A'' \vec{x}^{(k)} + \vec{x}_C^{\,\prime\prime(k)}.$$

Algoritmo 10: Metodo Adattivo Filtrato

---

function $\vec{x}^{(n)}$=FiltratoAPR()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}^{(k+1)} = A'' \vec{x}^{(k)} + \vec{x}_C^{\,\prime\prime(k)}$
  periodically,
  $[N, C]$ =SelezionaConvergenti$(\vec{x}^{(k)}, \vec{x}^{(k+1)}, \varepsilon)$;
  $[A'']$ =filter(A'', N, C)
  $[\vec{x}''_C]$ =filter$(\vec{x}^{(k)}, C)$
  periodically $\delta = \|A\vec{x}^{(k)} - \vec{x}^{(k)}\|_1$;
  $k = k + 1$;
until $\delta < \varepsilon$

3.2.1 Costo computazionale

La matrice A'' è molto più sparsa della matrice A, quindi la moltiplicazione $A''\vec{x}$ è molto meno costosa rispetto alla moltiplicazione $A\vec{x}$. Infatti il costo della prima moltiplicazione è analogo al costo di $A_N\vec{x}$.

3.3 Metodo adattivo modificato

È possibile ridurre ulteriormente il costo computazionale non ricalcolando le componenti del vettore di PageRank di pagine che stanno in N, ovvero che non convergono, collegate a pagine in C, che quindi sono già giunte a convergenza. Definiamo A_{NN} come i links dalla pagine che non convergono a pagine che non sono giunte a convergenza, A_{CC} sono i links delle pagine che convergono a pagine che non convergono.
Possiamo riscrivere la matrice A nel seguente:

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

quindi l'iterazione diventa:

$$\vec{x}_N^{(k+1)} = A_{NN}\vec{x}_N^{(k)} + A_{CN}\vec{x}_C^{(k)}.$$

Dato che $\vec{x}_C$ non cambia ad ogni iterazione anche la componente $A_{CN}\vec{x}_C^{(k)}$ non cambia ad ogni iterazione, perciò è necessario calcolare solamente $A_{CN}\vec{x}_C^{(k)}$ ogni volta che la matrice A è riordinata. Questa variante è esposta nell'algoritmo 11.

Algoritmo 11: Metodo Adattivo Modificato

---

function $\vec{x}^{(n)}$=ModificatoAPR()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}_N^{(k+1)} = A_{NN}\vec{x}_N^{(k)} + \vec{y}$;
  $\vec{x}_C^{(k+1)} = \vec{x}_C^{(k)}$;
  periodically,
  $[N, C]$ =SelezionaConvergenti$(\vec{x}^{(k)}, \vec{x}^{(k+1)}, \varepsilon)$;
  $\vec{y} = A_{CN}\vec{x}_C^{(k)}$;
  periodically $\delta = \|A\vec{x}^{(k)} - \vec{x}^{(k)}\|_1$;
  $k = k + 1$;
until $\delta < \varepsilon$

Non è necessario nell'implementazione esplicitare il riordinamento delle componenti: possiamo aumentare la sparsità di A_{CN} e A_{NN} per ridurre la loro grandezza effettiva.
L'algoritmo 12 è implementato basandosi sull'idea precedente, ci si può aspettare quindi che converga più velocemente dato che la somma parziale (denotata con $\vec{y}$) non è calcolata ad ogni iterazione.

Algoritmo 12: Metodo Adattivo Modificato Filtrato

function $\vec{x}^{(n)}$=FilterMAPR()
$\vec{x}^{(0)} = \vec{v}$;
$k = 1$;
repeat
  $\vec{x}^{(k+1)} = A_{NN}\vec{x}_N^{(k)} + \vec{y} + \vec{x}_C''$;
  periodically,
  N′=N; C′=C;
  $[N, C]$ =SelezionaConvergenti($\vec{x}^{(k)}, \vec{x}^{(k+1)}, \varepsilon$);
  $[A_{NN}'', A_{CN}'']$ =filter($A_{N'N'}'', A_{C'N'}'', N, C$)
  $[\vec{x}_C'']$ =filter($\vec{x}^{(k)}, C$)
  $\vec{y} = A_{CN}\vec{x}_C^{(k)}$;
  periodically, $\delta = \|A\vec{x}^{(k)} - \vec{x}^{(k)}\|_1$;
  $k = k + 1$;
until $\delta < \varepsilon$

3.4 Risultati sperimentali

È possibile che l'algoritmo erroneamente consideri che il pagerank di pagina sia giunto a convergenza quando, in realtà, non converge. Inoltre creare le nuove matrici riordinate ad ogni iterazione è molto costoso.
Per evitare questi problemi si utilizza l'algoritmo adattivo a fasi. In ciascuna fase iniziamo con la versione originale della struttura dei link, iteriamo un certo numero di volte, riduciamo la struttura dei link ed infine iteriamo un numero addizionale di volte. Nella fase successiva riduciamo il valore della tolleranza usato.
Questa strategia ha lo scopo di fare in modo che tutte le pagine abbiamo all'incirca lo stesso livello di errore, computando le iterate successive per raggiungere una specifica tolleranza finale.

Di seguito sono riportate, nella *tabella 1*, le principali differenze fra il metodo adattivo standard e il metodo adattivo filtrato al variare delle iterazioni per ogni fase (ipp). Osserviamo che per un numero maggiore di iterazioni per fase, non vi è una grande differenza fra i due metodi, al contrario di quanto succede per un numero di iterazioni più basso.

Adattivo	ipp = 5	ipp = 8	ipp = 20
numero di iterazioni	76	97	191
dimensione dell'insieme C	650153	530334	255867
tempo calcolo	0.82	0.78	0.60
Filter Adattivo	ipp = 5	ipp = 8	ipp = 20
numero di iterazioni	196	199	191
dimensione dell'insieme C	458723	450867	255609
tempo calcolo	2.37	1.91	0.87

Tabella 1: *Confronto dei due metodi al variare delle iterazioni per fase*

Se si considera la variazione del parametro α, come si può notare dalla *tabella 2*, il numero di iterazioni e il tempo di calcolo aumentano notevolmente nel caso del metodo adattivo filtrato; tuttavia il numero di links che non giungono a convergenza diminuisce.

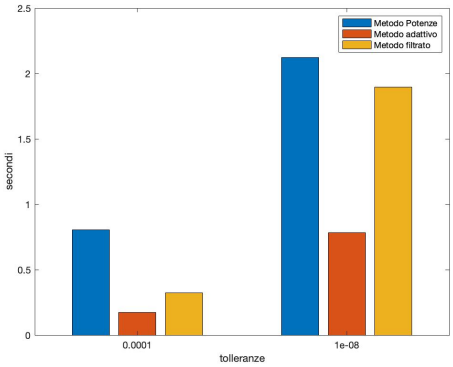
Adattivo	$\alpha = 0.50$	$\alpha = 0.85$	$\alpha = 0.99$
numero di iterazioni	50	97	918
dimensione dell'insieme C	246102	530334	661708
tempo calcolo	0.15	0.76	0.32
Filter-Adattivo	$\alpha = 0.50$	$\alpha = 0.85$	$\alpha = 0.99$
numero di iterazioni	50	199	2570
dimensione dell'insieme C	246102	450867	452237
tempo calcolo	0.30	1.87	24.75

Tabella 2: *Confronto dei due metodi al variare di* α

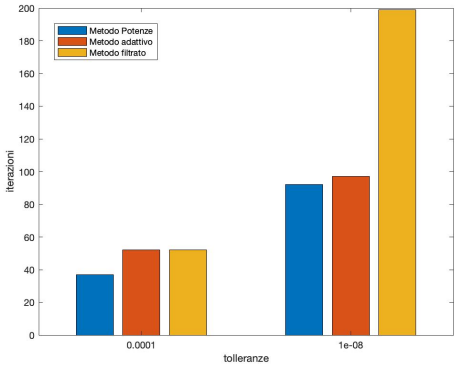
Consideriamo ora le principali differenze generate dal modificare il valore della tolleranza; come si può osservare dalla *tabella 3* si hanno risultati analoghi al caso precedente: aumentano sia il numero di iterazioni sia il tempo di calcolo e diminuisce la dimensione dell'insieme di links non convergenti.

Adattivo	tol=1e-3	tol=1e-8	tol=1e-13
numero di iterazioni	25	97	145
dimensione dell'insieme C	247913	530334	669911
tempo calcolo	0.07	0.77	1.46
Filter Adattivo	tol=1e-3	tol=1e-8	tol=1e-13
numero di iterazioni	25	199	315
dimensione dell'insieme C	247913	450867	486885
tempo calcolo	0.15	1.88	2.99

Tabella 3: *Confronto dei due metodi al variare della tolleranza*



(a) *Diversi tempi di calcolo*



(b) *Numero di iterazioni necessarie*

Figura 14: Confronto metodo potenze e metodi adattivi con diverse tolleranze

La *Figura 14* mostra invece, sempre al variare della tolleranza, un confronto anche con il metodo delle potenze. Si osserva quindi che il metodo delle potenze risulta avere un tempo di calcolo maggiore rispetto ai metodi adattivi, ed un numero di iterazioni per raggiungere la tolleranza desiderata minore. Il fatto che i metodi adattivi utilizzino un numero maggiore di iterazioni è dovuto al fatto che è necessario che in ogni fase le pagine abbiano all'incirca lo stesso livello di errore. Ciò nonostante, dato che il costo medio di iterazione diminuisce sensibilmente, si ha un miglioramento effettivo del metodo del PageRank.

Capitolo 4

Confronto tra i vari metodi considerati

In questo capitolo si vuole mettere a confronto tutti i metodi visti nei capitoli precedenti utilizzando diversi dataset.

4.1 Dati utilizzati

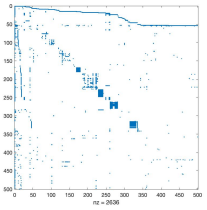
Presentiamo ora le diverse matrici utilizzate in questo studio:

- **Harvard Web**: matrice 500×500, che presenta 2636 non-zero,
- **Wiki vote**: matrice 8297×8297, che presenta 103689 non-zero,
- **Stanford Web**: matrice 281903×281903, che presenta 2312497 non-zero,
- **Google Web**: matrice 916428×916428, che presenta 916428 non-zero.

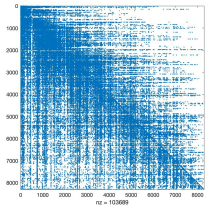
Inoltre utilizzeremo ancora la matrice **Stanford-Berkley** dei risultati precedenti. Di seguito, nella *Figura 15*, è riportato il grafico degli elementi non nulli delle matrici, si può osservare che le matrici Harvard Web e Wiki vote sono molto sparse mentre Stanford Web e Google Web sono più piene.

4.2 Confronto tra i metodi di estrapolazione

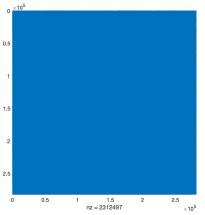
Lo scopo di questa sezione è fare un confronto fra i tre metodi di estrapolazione visti. In questi risultati si ha una tolleranza posta a 1e-8 e $\alpha = 0.85$; l'estrapolazione viene fatta, nel caso del metodo di estrapolazione di Aitken e epsilon, alla decima iterazione mentre per il metodo delle potenze ogni dieci iterazioni. Per quanto



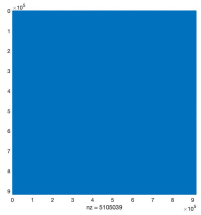
(a) *Matrice Harvard Web*



(b) *Matrice Wiki vote*



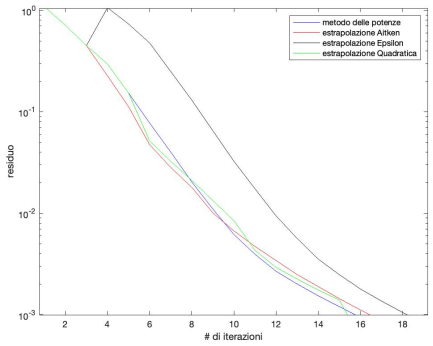
(c) *Matrice Stanford Web*



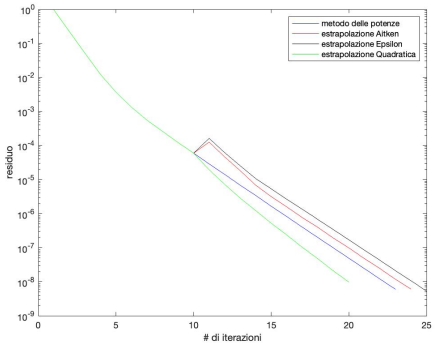
(d) *Matrice Google Web*

Figura 15: Grafico dei non-zero delle matrici

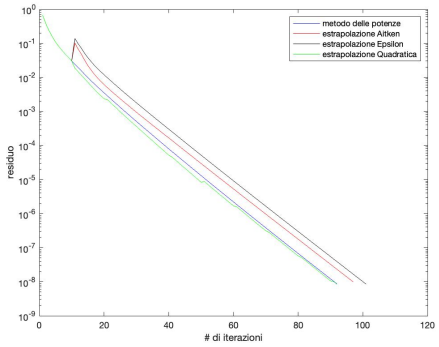
riguarda la matrice Harvard Web, date le dimensioni ridotte, si hanno risultati migliori se si pone tolleranza a 1e-3 e si applica l'estrapolazione alla terza iterata. Grazie alla buona approssimazione che si ha al momento dell'estrapolazione il metodo più efficiente per accelerare il calcolo del Pagerank risulta l'estrapolazione quadratica applicata periodicamente. Come si osserva dalla *Figura 16* infatti, indipendentemente dalla matrice scelta, l'estrapolazione quadratica converge più velocemente.
Le estrapolazioni di Aitken e epsilon invece non migliorano in tutti i casi i risultati del calcolo del metodo delle potenze, infatti, come visto nel capitolo precedente, a causa dell'approssimazione più grossolana si ha un picco iniziale del residuo che rallenta la convergenza.



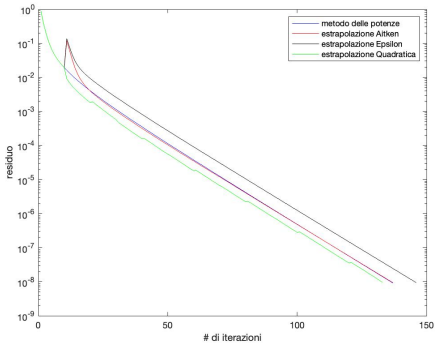
(a) *Matrice Harvard Web*



(b) *Matrice Wiki vote*



(c) *Matrice Stanford Web*



(d) *Matrice Google Web*

Figura 16: Confronto metodi di estrapolazione

4.3 Confronto tra i metodi di estrapolazione e metodi adattivi

Sono di seguito riportate due tabelle che mettono a confronto il numero di iterazioni e il tempo di calcolo utilizzato nei diversi metodi visti. Come si può osservare, ad eccezione della matrice Harvard Web, i metodi adattivi, soprattutto quello filtrato, utilizzano un numero di iterazioni più elevato; questo poiché i metodi adattivi richiedono che tutti i nodi giungano alla tolleranza fissata prima di ridurre il valore soglia tl e applicare l'algoritmo all'intera matrice.
I metodi adattivi risultano meno efficaci per matrici di piccole dimensioni, questo poiché hanno un overhead maggiore rispetto ai metodi di estrapolazione. Per esempio nel metodo adattivo ad ogni fase si creano gli insiemi C ed N e si riordina il vettore controllando quali nodi convergono.
Invece in generale, per quanto riguarda matrici di dimensioni maggiori, si osserva un miglioramento del metodo adattivo soprattutto per quanto riguarda il tempo di calcolo.

Dataset	**Potenze**	**Aitken**	**Epsilon**	**Quadratica**	**Adattivo**	**Filtrato**
H	70	11	11	62	**5**	**5**
W	23	24	25	**22**	52	53
S	**92**	97	101	109	98	220
SB	92	99	102	**88**	97	199
G	89	**87**	95	88	97	223

Tabella 4: *Confronto delle iterazioni con* $\alpha = 0.85$

Dataset	**Potenze**	**Aitken**	**Epsilon**	**Quadratica**	**Adattivo**	**Filtrato**
H	0.0019	0.0074	0.0081	0.0125	**0.0010**	0.0028
W	**0.0070**	0.0127	0.0166	0.0131	0.0084	0.0165
S	1.0195	1.0873	1.1396	1.2370	**0.4887**	1.3346
SB	1.9723	2.4121	2.4809	2.2845	**0.7513**	1.9784
G	3.0714	2.9476	3.0818	3.0591	**1.2302**	4.1756

Tabella 5: *Confronto tempi di convergenza*

4.4 Conclusioni

In questa tesi abbiamo analizzato l'algoritmo del PageRank, ovvero un algoritmo che analizza l'importanza dei links basandosi sull'idea che ebbero Brin e Page nel

1998. Inizialmente è stato presentato l'algoritmo del metodo delle potenze, nonostante questo algoritmo sia semplice è alla base della ricerca del PageRank.
Abbiamo poi discusso tre metodi di estrapolazione che accelerano l'algoritmo del metodo delle potenze; in particolare l'estrapolazione quadratica, applicata periodicamente, approssima in modo più preciso il valore di PageRank in tutti i casi visti. Approssimando, infatti, di con tre autovettori al posto di due (come avviene negli altri due metodi) si ottengono risultati migliori in termini di tempo di calcolo e velocità di convergenza.
Infine, abbiamo studiato i metodi adattivi: questi si basano sull'idea di non ricalcolare il PageRank di pagine che sono già giunte a convergenza. Come visto in precedenza questi metodi richiedono un numero di iterazioni abbastanza elevato. D'altra parte, soprattutto per matrici di grandi dimensioni, migliorano il metodo delle potenze, nonostante abbiano un maggior costo computazionale.

Bibliografia

[1] Kamvar, S.D., Haveliwala, T.H., Manning, C.D., Golub, G.H., *Extrapolation methods for accelerating PageRank computations,* Proceedings of the 12th international conference on World Wide Web, 2003.

[2] Kamvar, S.D., Haveliwala, T.H., Golub, G.H., *Adaptive methods for the computation of PageRank*, Linear Algebra and its Applications 386, 2004, p. 51-65.

[3] Langville,A.N., Meyer,C.D., *Deeper inside pagerank*, Internet Mathematics 1.3, 2004, p.335-380.

[4] Quarteroni, A., Sacco,R., Saleri,F., *Matematica Numerica*, Springer,2008.

[5] Simoncini, V., *Dispense del corso Matematica Computazionale.*