Seminario di "Metodi Numerici per Catene di Markov": Aggiornamento del Pagerank di *Google* e cenni sull' *ordinal rank*

Mele Giampaolo

18 dicembre 2011

Sommario

Questo documento contiene il seminario presentato per l'esame di "Metodi numerici per catene di Markov". Sarà affrontato il problema del pagerank di google, in particolare l'aggiornamento del vettore di probabilità di una catena di markov dopo un update, dove l'update prevede l'aggiunta/rimozione di alcuni stati e la modifica della matrice di transizione e sarà dato un cenno all'ordinal rank. Sarà infine presentato l'EigenTrust che è un modello simile a quello di google utilizzato nelle reti p2p.

1 Introduzione

1.1 Descrizione del modello Google

In questa sezione si darà una descrizione minimale del modello di ricerca di google, i dettagli si trovano ad esempio in [1] oppure sui primi capitoli di [2].

Si consideri il seguete grafo: l'insieme dei vertici V è costituito da tutte le pagine web mentre l'insieme degli archi E è costituito da tutti i link, ovvero c'è un arco tra il vertice i e il vertice j se c'è un link nella pagina i alla pagina j. D'ora in avanti si indicherà con $\boldsymbol{H}$ la matrice di adiacenza di questo grafo (del web) e con n la sua dimensione, ovvero

$$\boldsymbol{H} = (h_{i,j}) \qquad \text{dove} \qquad h_{i,j} = \begin{cases} 1 & \text{se} \quad (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Come ben si può immaginare $\boldsymbol{H}$ ha molte proprietà, le più importanti sono

- dimensione elevata, il numero di pagine web è $n = O(10^{10})$
- sparsità
- presenza di cluster

Il modello matematico sul quale si basa la ricerca di google ha diverse interpretazioni, in questo contesto fa comodo tenere a mente solo quella probabilistica. Si supponga di voler fare una ricerca, si dovranno dunque cercare i siti che contengono determinate parole, daltronde scrivendo " Cinema Pisa " è preferibile che è i primi risultati siano gli indirizzi internet dei cinema presenti a Pisa, dopo eventuali palinsesti e infine recensioni e simili; di certo non si vorrà tra i primi risultati un articolo che spiega perchè a Pisa la gente preferisce andare al cinema piuttosto che a teatro. Quindi sarà necessario ordinare

i risultati seguendo un certo criterio, questa operazione è nota come pagerank.

Si consideri un utente che si comporti seguendo l'idea di "random walk", ovvero parte dalla pagina i_1 e da qui clicca su un link a caso e arriva alla pagina i_2, ancora da qui clicca su un link a caso .... Dove si troverà dopo n click? Per rispondere a questa domanda è necessario introdurre la matrice $\boldsymbol{G}$ costruita normalizzando rispetto la norma 1 le righe della matrice $\boldsymbol{H}$, ovvero

$$\boldsymbol{G} = (g_{i,j}) \qquad \text{dove} \qquad g_{i,j} = \begin{cases} \frac{h_{i,j}}{\sum_{j=1}^{n} h_{i.j}} & \text{se} \quad (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Quindi $g_{i,j}$ è la probabilità un utente che si trova nella pagina i si sposti nella pagina j. Si osserva che $\boldsymbol{G}$ è una matrice stocastica, ovvero $\boldsymbol{Ge} = \boldsymbol{e}$, quindi in analogia a random walk, rispondere alla domanda equivale a trovare il vettore stazionario di probabilità, ovvero $\boldsymbol{\pi}$ tale che $\boldsymbol{\pi}^T\boldsymbol{G} = \boldsymbol{\pi}^T$ con $\|\boldsymbol{\pi}\|_1 = 1$.

Si presenta subito un problema: per come si è formulato il problema questo vettore di probabilità non è unico. Infatti si dimostra che l'unicità di tale vettore dipende dall'irriducibilità della matrice, daltronde non è difficile immaginare che $\boldsymbol{G}$ sia tutt'altro che irriducibile, pertanto sarà necessario modificare il modello per rendere consistente il problema.

Dire che l'utente clicchi a caso sui link di una pagina è limitativo, più realistico è supporre che l'utente possa anche decidere di visitare una pagina non linkata dalla pagina corrente, ad esempio si trova su un sito sportivo e decide di voler controllare le email. Un primo modo per fare questa correzione è quello di effettuare una correzione di rango 1 alla matrice $\boldsymbol{G}$, ad esempio si può pensare che l'utente possa cambiare pagina e digitare un indirizzo a caso, ciò equivale a fare una combinazione convessa e considerare

$$\boldsymbol{A} = \gamma\boldsymbol{G} + (1-\gamma)\frac{\boldsymbol{e}\cdot\boldsymbol{e}^T}{n}$$

Daltronde l'utente non cambierà pagina a caso quindi non è sensato dare lo stesso peso a tutte le pagine, se ad esempio questi controlla ogni giorno le emai è probabile che cambi pagina e vada a controllare la posta, oppure se la sua età è compresa tra i 16 e i 25 anni è probabile che cerchi notizie sportive o recensioni di film. Quindi è necessario introdurre un vettore che pesi le pagine web basandosi sulla storia delle precedenti navigazioni dell'utente (siti visitati frequentemente) e con le informazioni note (sesso, età, preferenze musicali, locazione geografica ...), tale vettore $\boldsymbol{v}$ è detto vettore di personalizzazione (tutt'ora vi è un acceso dibattito etico su quanto sia giusto usare questa correzione), quindi si considererà d'ora in avanti

$$\boldsymbol{A} = \gamma\boldsymbol{G} + (1-\gamma)\boldsymbol{e}\cdot\boldsymbol{v}^T$$

Dove appunto $v(i)$ è la probabilità che l'utente si sposti sulla pagina i (questa probabilità sarà ad esempio tanto più altra quante più volte questa pagina è stata visualizzata) segue dunque che $\|\boldsymbol{v}\|_1 = 1$, quindi nuovamente $\boldsymbol{A}$ è combinazione convessa di matrici stocastiche quindi è stocastica.

Non è ora difficile vedere che a meno di scegliere nel modo opportuno $\boldsymbol{v}$ si ha che $\boldsymbol{A}$ è irriducibile, ad

esempio se $\boldsymbol{v} > \mathbf{0}$ il grafo associato ad $\boldsymbol{A}$ è non orientato e connesso; chiedere $\boldsymbol{v} > 0$ è restrittivo ma seguendo la stessa idea si può ad esempio chiedere che $\boldsymbol{v}$ renda cominicanti tutti i nodi del grafo (non è il caso in questo contesto di approfondire la questione per rendere scorrevole la lettura). Ad ogni modo si è giunti ad avere una matrice stocastica irriducibile, dunque il vettore di probabilità esiste ed è unico. Per concludere la descrizione del modello di ricerca di google si osserva che γ indica la probabilità che l'utente si sposti dalla pagina i ad una pagina non linkata da i, google ha settato questo parametro a $\gamma = 0.85$ ma in altri contesti (algoritmo dell'eigentrust) potrebbe esser conveniente cambiare questo parametro (che influisce sulla velocità di convergenza dei metodi per il calcolo del vettore di probabilità). Calcolato il vettore delle probabilità $\boldsymbol{\pi}$ è sensato affermare che la pagina i è più importante della pagina j se $\pi(i) > \pi(j)$, ovvero se è più probabile che l'utente dopo alcune ore di navigazione si trovi nella pagina i piuttosto che nella pagina j.

2 Update del vettore di probabilità

Tutto quello che riguarda l'update del vettore di probabilità è tratto da [3].

Sia $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ matrice di transizione di una catena di Markov omogenea e irriducibile con m stati e sia $\boldsymbol{\phi}^T = (\phi_1, \ldots, \phi_m)$ il vettore di probabilità. Si supponga di fare l'update (aggiornamento) della catena di Markov, le operazioni previste da un update sono:

- Aggiunta/rimozione di stati
- Alterazione degli elementi della matrice di transizione delle probabilità

L'ultimo punto va inteso nel seguente modo: se i e j sono stati che esistevano anche prima l'update allora è possibile cambiare il valore che esprime la probabilità di passare da i a j, se uno dei due non esisteva prima dell'update allora quella quantità deve esser definita dall'update stesso.

Dopo l'update si avrà una nuova matrice di transizione $\boldsymbol{P} \in \mathbb{R}^{n\times n}$, si chiederà che questa sia irriducibile.

La domanda alla quale si cercherà di rispondere è: come fare l'update del vettore di probabilità? Ovvero, come trovare $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ vettore di probabilità conoscendo $\boldsymbol{\phi}$?

Affinchè la domanda abbia senso è necessario contestualizzare il problema e fare ulteriori ipotesi.

Si assuma che $\boldsymbol{Q}$ sia la matrice di transizione di google (quella che nella sezione precedente era denotata con $\boldsymbol{A}$), ogni giorno vengono create ed eliminate migliaia di pagine e al tempo stesso creati ed eliminati migliaia di link, quindi la matrice di transizione cambierà continuamente; google calcola il pagerank circa una volta ogni tre settimane, daltronde numero di pagine web è $O(10^9)$, pertanto anche se ad ogni update ci sono state migliaia di modifiche in generale il vettore del pagerank non sarà cambiato di molto.
Date le dimensioni del problema, non è realistico pensare che ogni volta si debba fare il calcolo completo del pagerank, è preferibile cercare un modo per ottenere il pagerank nuovo a partire dal vecchio, quindi sfruttare la conoscenza di $\boldsymbol{\phi}$ per calcolare $\boldsymbol{\pi}$. Ci si aspetta infatti che se $\boldsymbol{Q}$ è vicina a $\boldsymbol{P}$ allora $\boldsymbol{\phi}$ è vicino a $\boldsymbol{\pi}$.

Quando si afferma che $\boldsymbol{\phi}$ è vicino a $\boldsymbol{\pi}$ si intende che se una certa pagina i aveva un certo valore di rank $\phi(i)$ allora il nuovo valore di rank $\pi(i)$ sarà molto vicino al precedente, se invece la pagina è

stata eliminata allora si intenderà che il nuovo valore di rank sarà nullo, quindi $\phi(i)$ era molto piccolo. Si osserva che quanto appena detto è in accordo al modello creato, infatti le pagine web che vengono cancellate sicuramente non sono "importanti", difficilmente sarà cancellata una pagina con un alto rank. Lo stesso discorso si ripete su $\boldsymbol{Q}$ e $\boldsymbol{P}$, ovvero queste le intenderemo vicine se gli stati che si sono aggiunti o cancellati che hanno poca "importanza", ovvero $\phi(i)$ è piccolo oppure se $\phi(i)$ è vicino $\pi(i)$. Questi discorsi sono tutti sensati nel caso di google.

Esiste un algoritmo che permette di eseguire l'update esatto del vettore di probabilità, daltronde per il calcolo di quest'ultimo si utilizza il metodo delle potenze quindi già $\boldsymbol{\phi}$ sarà un'approssimazione del vettore di probabilità di $\boldsymbol{Q}$ quindi non è sensato chiedere l'update esatto, senza contare che il costo computazionale è elevato quindi non è utilizzabile per il problema di google. Pertanto sarà mostrato solo l'approccio iterativo per avere l'update approssimato.

2.1 Primi tentativi

Come si è già detto per il calcolo del vettore di probabilità si utilizza il metodo delle potenze che consiste nell'iterare

$$\begin{cases} \boldsymbol{\phi}^{(j+1)^T} = \boldsymbol{\phi}^{(j)^T}\boldsymbol{Q} \\ \boldsymbol{\phi}^{(0)^T} = \boldsymbol{x}^T \end{cases}$$

Sotto le ipotesi fatte (irriducibilità) si può dimostrare che $\boldsymbol{\phi}^{(n)}$ converge a $\boldsymbol{\phi}$ per ogni scelta di $\boldsymbol{x}$ e il numero di iterazioni necessarie per avere una buona approssimazione dipende da due fattori: il secondo autovalore dominante di $\boldsymbol{Q}$ e da quando il punto iniziale $\boldsymbol{x}$ è vicino $\boldsymbol{\phi}$.

Per comodità si consideri invariato il numero di stati nell'update (non è difficile generalizzare), allora una prima strategia a cui si può pensare è quella di utilizzare il metodo delle potenze partendo da ϕ che per ipotesi è vicino π, quindi

$$\begin{cases} \boldsymbol{\pi}^{(j+1)^T} = \boldsymbol{\pi}^{(j)^T}\boldsymbol{P} \\ \boldsymbol{\pi}^{(0)^T} = \boldsymbol{\phi}^T \end{cases} \tag{1}$$

Daltronde questo metodo è inefficace, per capirne il motivo è necessario un richiamo.

2.1.1 Tasso asintotico di convergenza del metodo delle potenze

Si definisca il tasso asintotico di convergenza come

$$R = -\log_{10}(|\lambda_2|)$$

Dove λ_2 è il secondo autovalore più grande di $\boldsymbol{P}$. Questo numero dice indicativamente quante cifre significative bisogna aspettarsi dopo un certo numero di iterate, infatti supponendo che l'errore al passo k sia $\epsilon(k)$, allora è noto che

$$\left|\frac{\epsilon(k-1)}{\epsilon(k)}\right| \simeq \frac{1}{|\lambda_2|}$$

Non è difficile intuire che tale relazione valga componente per componente

$$\left|\frac{\epsilon_i(k-1)}{\epsilon_i(k)}\right| \simeq \frac{1}{|\lambda_2|}$$

Si supponga ad esempio

$$|\epsilon_i(k-1)| = 10^{-q} \quad \text{e} \quad |\epsilon_i(k)| = 10^{-p} \quad \text{con} \quad 0 < q \leq p$$

Questo vuol dire che si hanno q cifre esatte nella $(i-1)$-esima iterata e p in quella successiva.

Segue che

$$p - q = \log_{10} \left| \frac{\epsilon_i(k-1)}{\epsilon_i(k)} \right| \simeq -\log_{10}(|\lambda_2|)$$

Questo semplice ragionamento mostra che il numero di cifre esatte calcolate con il metodo delle potenze dipende dal tasso asintotico di convergenza R, quindi (dopo un certo numero di passi) ad ogni $1/R$ iterate si otterrà una cifra significativa. Si userà questo per mostrare che il metodo appena esposto per l'update del pagerank non funziona bene. I dettagli su questa argomentazione si trovano in [4] (pagina 621).

Si riconsideri dunque metodo (1) per il calcolo dell'update, si supponga $\|\boldsymbol{P} - \boldsymbol{Q}\|$ abbastanza piccolo da assicurare che $\boldsymbol{\pi}$ e $\boldsymbol{\phi}$ abbiano la prima cifra significativa uguale, si vuole calcolare con (1) le prime dodici cifre significative di $\boldsymbol{\pi}$. Avendo già la prima cifra significativa, basterà fare $11/R$ iterate, se avessimo preso $\boldsymbol{\pi}^{(0)}$ qualsiasi sarebbero stare necessarie $12/R$ iterate, quindi in questo caso si ha un rispormio dell 8%. Se ad esempio $\lambda_2 = 0.85$ allora si ha che fare l'update del vettore di probabilità costa 156 iterazioni (supponendo di volere le prime 12 cifre significative) mentre calcolarlo con il semplice metodo delle potenze richiede 171 quindi si risparmiano circa 15 iterazioni.

In conclusione (1) non è il metodo ottimale per l'update del vettore di probabilità.

2.2 Metodi di aggregazione

Ai fini del calcolo del vettore di probabilità si possono utilizzare diversi algoritmi in base al tipo di problema da risolvere. I metodi che si stanno per mostrare sono detti metodi di aggregazione e l'idea che è alla base di questi ultimi è quella di diminuire il numero di stati di una catena di Markov aggregandoli e considerando dei macrostati.

2.2.1 Aggregazione esatta

L'obbiettivo di questa sottosezione è richiamare alcuni risultati sull'agregazione esatta che saranno utili per giustificare l'algoritmo di aggregazione iterata il quale esegue l'update del vettore di probabilità

Si consideri una catena di Markov finita e irriducibile con S insieme degli stati data una sua partizione $S = G_1 \cup G_2 \cup \cdots \cup G_k$ si ha una partizione a blocchi delle matrice di transizione (con blocchi diagonali quadrati)

$$\boldsymbol{P} = \begin{array}{c} \\ G_1 \\ G_2 \\ \vdots \\ G_k \end{array} \begin{array}{c} \begin{array}{cccc} G_1 & G_2 & \dots & G_k \end{array} \\ \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} & \dots & \boldsymbol{P}_{1,k} \\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} & \dots & \boldsymbol{P}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_{k,1} & \boldsymbol{P}_{k,2} & \dots & \boldsymbol{P}_{k,k} \end{pmatrix} \end{array} \tag{2}$$

Dunque si avrà una *catena di Markov madre* formata dagli stati $G_1, G_2, \ldots, G_k$, che sono anche detti *macrostati*, nella catena madre non interessano le transizioni interne ai macrostati G_i ma solo le transizioni da un macrostato all'altro, quindi la catena di markov madre avrà una certa matrice di transizione che

si descriverà a breve.

La catena di Markov madre induce k catene più piccole dette *catene di Markov censurate*; nella i-esima catena di markov censurata si avranno come stati gli elementi di G_i e si considereranno solo le transizioni tra gli elementi di G_i e non quelle esterne, questo giustifica la terminologia utilizzata, ovvero perchè queste catene sono dette censurate.

Non è difficile provare che la matrice di transizione delle probabilità per l'i-esima catena di markov censurata (detta anche i-esimo complemento stocastico) è la seguente

$$\boldsymbol{S}_i = \boldsymbol{P}_{i,i} + \boldsymbol{P}_{i,\star}(\boldsymbol{I} - \boldsymbol{P}_i^{\star})^{-1}\boldsymbol{P}_{\star,i}$$

Dove $\boldsymbol{P}_{i,\star}$ e $\boldsymbol{P}_{\star,i}$ sono rispettivamente l'i-esima riga e l'i-esima colonna di blocchi senza però $\boldsymbol{P}_{i,i}$, mentre $\boldsymbol{P}_i^{\star}$ è la sottomatrice di $\boldsymbol{P}$ ottenuta togliendo l'i-esima riga e l'i-esima colonna di blocchi. Si può mostrare che se la catena di markov è irriducibile allora lo sono anche le censurate.

Esempio 2.1. Si supponga di avere una catena di markov e di voler aggregare gli stati in soli due macrostati $S = G_1 \cup G_2$, allora le si avrà

$$\boldsymbol{P} = \begin{array}{c} \\ G_1 \\ G_2 \end{array} \begin{array}{c} \begin{array}{cc} G_1 & G_2 \end{array} \\ \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} \\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} \end{pmatrix} \end{array}$$

In questo caso le matrici di transizione delle catene censurate sono rispettivamente

$$\boldsymbol{S}_1 = \boldsymbol{P}_{1,1} + \boldsymbol{P}_{1,2}(\boldsymbol{I} - \boldsymbol{P}_{2,2})^{-1}\boldsymbol{P}_{2,1} \qquad \boldsymbol{S}_1 = \boldsymbol{P}_{2,2} + \boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{P}_{1,2}$$

Proposizione 2.1. Se $\boldsymbol{\pi}^T = (\boldsymbol{\pi}_1^T | \dots | \boldsymbol{\pi}_k^T)$ è il vettore di probabilità di $\boldsymbol{P}$ dove la dimensione di $\boldsymbol{\pi}_i$ è la stessa di $\boldsymbol{P}_{i,i}$, allora si ha che il vettore di probabilità dell'i-esima catena di Markov censurata è il seguente

$$\boldsymbol{s}_i^T = \frac{\boldsymbol{\pi}_i^T}{\boldsymbol{\pi}_i^T \cdot \boldsymbol{e}}$$

A questo punto è possibile esibire la matrice di transizione per la catena madre, questa è detta anche matrice di transizione aggregata, ed è la seguente

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{s}_1^T\boldsymbol{P}_{1,1}\boldsymbol{e} & \dots & \boldsymbol{s}_1^T\boldsymbol{P}_{1,k}\boldsymbol{e} \\ \vdots & \ddots & \vdots \\ \boldsymbol{s}_k^T\boldsymbol{P}_{k,1}\boldsymbol{e} & \dots & \boldsymbol{s}_k^T\boldsymbol{P}_{k,k}\boldsymbol{e} \end{pmatrix} \tag{3}$$

Remark 2.1. Si osservi che $\boldsymbol{A} \in \mathbb{R}^{k\times k}$, quindi, per quando detto all'inizio $(\boldsymbol{A})_{i,j} = \boldsymbol{s}_i^T\boldsymbol{P}_{i,j}\boldsymbol{e}$ è la probabilità che si passi dal macrostato G_i al macrostato G_j. Si può mostrare che se la catena di markov è irriducibile allora anche lo è anche la catena aggregata, quindi $\boldsymbol{A}$ è irriducibile.

Teorema 2.1. Data una catena di markov irriducibile e una partizione degli stati $S = G_1 \cup \dots \cup G_k$, sia $\boldsymbol{P}$ la matrice di transizione partizionata a blocchi (2) con vettore di probabilità

$$\boldsymbol{\pi}^T = (\boldsymbol{\pi}_1^T | \dots | \boldsymbol{\pi}_k^T) \qquad \text{(partizionato coerentemente con la partizione di } \boldsymbol{P})$$

Sia $\boldsymbol{A}$ la matrice di transizione aggregata (3) e sia $\boldsymbol{\alpha}^T = (\alpha_1, \ldots, \alpha_k)$ il suo vettore di probabilità, allora valgono

$$\begin{aligned} \alpha_i &= \boldsymbol{\pi}_i^T \cdot \boldsymbol{e} \\ \boldsymbol{\pi}^T &= (\alpha_1 \boldsymbol{s}_1^T, \alpha_2 \boldsymbol{s}_2^T, \ldots, \alpha_k \boldsymbol{s}_k^T) \end{aligned} \tag{4}$$

Ora è possibile mostrare l'algoritmo per il calcolo del vettore di probabilità con aggregazione esatta:

1 - Aggregazione degli stati $S = G_1 \cup \cdots \cup G_k$

2 - Partizionamento a blocchi della matrice di transizione (2)

3 - Calcolo del complementi stocastici $\boldsymbol{S}_i$

4 - Calcolo dei vettori di probabilità $\boldsymbol{s}_i$ delle catene censurate

5 - Calcolo della matrice aggregata (3)

6 - Calcolo del vettore di probabilità della matrice aggregata

7 - Ricostruzione del vettore di probabilità della catena originaria con la (4)

Conclusioni

Si conclude con delle osservazioni: innanzitutto si è ridotto il problema del calcolo del vettore di probabilità di una catena di markov con $|S| = m$ stati a quello del calcolo di k vettori di probabilità di k catene di markov (censurate) con meno stati e al calcolo di un vettore di probabilità di una catena di markov con k stati (catena aggregata). Sembrerebbe esserci un notevole risparmio dal punto di vista del costo computazionale ma non è così, infatti il calcolo dei complementi stocastici prevede l'inversione di una matrice; quindi questo approccio è utile ad esempio se le aggregazioni si presentano in modo naturale (la matrice di transizione è diagonale a blocchi) o quasi (problemi NCD) e questo accade nel caso di google (si era parlato di presenza di cluster). Un'altra strada potrebbe esser l'approssimazione dei complementi stocastici oppure si può provare direttamente ad approssimare dei vettori di proabilità delle catene censurate ovvero gli $\boldsymbol{s}_i$. Si seguirà quest'ultima strada, nota come aggregazione approssimata, per l'update del vettore di probabilità.

2.2.2 Aggregazione approssimata

Si mostrerà ora come eseguire l'upload del vettore di probabilità utilizzando l'aggregazione approssimata. Brevemente si ricorda il problema: data una catena di markov con una matrice di transizione $\boldsymbol{Q}$ e vettore di proabilità $\boldsymbol{\phi}$, l'update della catena di markov consiste nell'aggiungere/rimuovere stati e modificare la matrice di transizione, quindi la catena di markov dopo l'update avrà matrice di transizione $\boldsymbol{P}$ e vettore di probabilità $\boldsymbol{\pi}$. L'obbiettivo è calcolare $\boldsymbol{\pi}$ supponendo noto $\boldsymbol{\phi}$.

Si consideri ora la catena di markov dopo l'update, allora si partizioni l'insieme degli stati $S = G \cup \overline{G}$ seguendo il seguente criterio: G è l'insieme degli stati "più affetti da cambiamenti" (in un senso che sarà chiaro più avanti) e dei nuovi stati, mentre $\overline{G} = S \setminus G$, si sottolinea che non si stanno aggregando gli stati, per ora si è solo fatto un partizionamento.

Dato che la perturbazione dovuta all'update coinvolge solo alcuni stati, nel caso di catene di markov di grandi dimensioni e sparse (come nel caso di google), l'intuizione porta ad affermare che la maggior

parte delle componenti del vettore di probabilità non cambierà troppo, quindi dopo l'update solo alcune componenti saranno cambiate, ovvero la modifica sarà locale.

Pertanto si avrà il seguente partizionamento della matrice di transizione

$$\boldsymbol{P} = \begin{array}{c} \\ G \\ \overline{G} \end{array} \begin{array}{c} \begin{array}{cc} G & \overline{G} \end{array} \\ \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} \\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} \end{pmatrix} \end{array}$$

Dove $\boldsymbol{P}_{1,1} \in \mathbb{R}^{g \times g}$ con $g = |G|$ e di conseguenza $\boldsymbol{P}_{2,2} \in \mathbb{R}^{(n-g)\times(n-g)}$, quindi allo stesso modo si avrà un partizionamento del vettore di probabilità

$$\boldsymbol{\pi}^T = (\pi_1, \dots, \pi_g, \pi_{g+1}, \dots, \pi_n) = (\pi_1, \dots, \pi_g | \overline{\boldsymbol{\pi}}^T)$$

A questo punto si procede con l'aggregazione, si aggrega $\overline{G}$ in un unico macrostato e gli altri si lasciano invariati. Quindi in analogia a quanto detto nella sezione precedente si avranno $g+1$ catene di markov censurate, dove g sono costituite da un unico stato (quindi non c'è da dir nulla), e poi ci sarà quella di $\overline{G}$.

A questo punto se volessimo seguire l'algoritmo di aggregazione esatta mostrato nella sezione precedente si avrebbe

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} \\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} \end{pmatrix} = \left(\begin{array}{c|c|c|c} p_{1,1} & \dots & p_{1,g} & \boldsymbol{P}_{1,\star} \\ \hline \vdots & \ddots & \vdots & \vdots \\ \hline p_{g,1} & \dots & p_{g,g} & \boldsymbol{P}_{g,\star} \\ \hline \boldsymbol{P}_{\star,1} & \dots & \boldsymbol{P}_{\star,g} & \boldsymbol{P}_{2,2} \end{array}\right)$$

dove appunto si è posto

$$\boldsymbol{P}_{1,1} = \begin{pmatrix} p_{1,1} & \dots & p_{1,g} \\ \vdots & \ddots & \vdots \\ p_{g,1} & \dots & p_{g,g} \end{pmatrix} \quad , \quad \boldsymbol{P}_{1,2} = \begin{pmatrix} \boldsymbol{P}_{1,\star} \\ \vdots \\ \boldsymbol{P}_{g,\star} \end{pmatrix} \quad , \quad \boldsymbol{P}_{2,1} = \begin{pmatrix} \boldsymbol{P}_{\star,1} & \dots & \boldsymbol{P}_{\star,g} \end{pmatrix}$$

Quindi si ha che la matrice di aggregazione esatta avrà ordine $g+1$ e sarà

$$A = \left(\begin{array}{c|c|c|c} p_{1,1} & \dots & p_{1,g} & \boldsymbol{P}_{1,\star}\boldsymbol{e} \\ \hline \vdots & \ddots & \vdots & \vdots \\ \hline p_{g,1} & \dots & p_{g,g} & \boldsymbol{P}_{g,\star}\boldsymbol{e} \\ \hline \boldsymbol{s}^T\boldsymbol{P}_{\star,1} & \dots & \boldsymbol{s}^T\boldsymbol{P}\star, g & \boldsymbol{s}^T\boldsymbol{P}_{2,2}\boldsymbol{e} \end{array}\right)$$

Dove $\boldsymbol{s}$ è il vettore di probabilità della catena censurata non banale che ha come matrice di transizione

$$\boldsymbol{S} = \boldsymbol{P}_{2,2} + \boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{P}_{1,2}$$

Quindi è possibile riscrivere $\boldsymbol{A}$ (sfruttando la stocasticità) in forma stringata come

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2}\boldsymbol{e} \\ \boldsymbol{s}^T\boldsymbol{P}_{2,1} & \boldsymbol{s}^T\boldsymbol{P}_{2,2}\boldsymbol{e} \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2}\boldsymbol{e} \\ \boldsymbol{s}^T\boldsymbol{P}_{2,1} & 1 - \boldsymbol{s}^T\boldsymbol{P}_{2,1}\boldsymbol{e} \end{pmatrix}$$

Se il vettore di probabilità di $\boldsymbol{A}$ è

$$\boldsymbol{\alpha}^T = (\alpha_1, \ldots, \alpha_g, \alpha_{g+1})$$

Allora usando il teorema 2.1 il vettore di probabilità esatto relativo a $\boldsymbol{P}$ sarà

$$\boldsymbol{\pi}^T = (\pi_1, \ldots, \pi_g, \pi_{g+1}, \ldots, \pi_n) = (\pi_1, \ldots, \pi_g | \overline{\boldsymbol{\pi}}^T) = (\alpha_1, \ldots, \alpha_g | \alpha_{g+1} \boldsymbol{s}^T) \tag{5}$$

Daltronde all'inizio si è detto che è sufficiente una approssimazione del vettore di probabilità della catena censurata, in particolare dato che questa è costituita dagli stati $\overline{G}$ che hanno ricevuto una modifica minima dall'update, allora $\overline{\boldsymbol{\phi}} \simeq \overline{\boldsymbol{\pi}}$, è allora sensato approssimare $\tilde{s} = \overline{\boldsymbol{\phi}}/(\overline{\boldsymbol{\phi}}^T \boldsymbol{e})$ quindi si definisce

$$\boldsymbol{\delta}^T = \boldsymbol{s}^T - \tilde{\boldsymbol{s}}^T = \frac{\overline{\boldsymbol{\pi}}^T}{\overline{\boldsymbol{\pi}}^T \boldsymbol{e}} - \frac{\overline{\boldsymbol{\phi}}^T}{\overline{\boldsymbol{\phi}}^T \boldsymbol{e}}$$

$$\boldsymbol{E} = \boldsymbol{A} - \tilde{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\delta}^T \boldsymbol{P}_{2,1} & -\boldsymbol{\delta}^T \boldsymbol{P}_{2,1} \boldsymbol{e} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\delta}^T \end{pmatrix} \boldsymbol{P}_{2,1} \left(\begin{array}{c|c} \boldsymbol{I} & -\boldsymbol{e} \end{array} \right)$$

Quindi se $\boldsymbol{\delta}$ è abbastanza piccolo certamente si potrà avere una approssimazione buona di $\boldsymbol{\pi}$. Daltronde si può mostrare che in alcuni casi, ad esempio se il secondo autovalore dominante è vicino ad 1, l'approssimazione può non esser buona. Allora l'idea è di iterare questo algoritmo e fare in modo che questo sia solo il primo passo di un algoritmo iterativo.

2.2.3 Aggregazione iterata

Il metodo che sta per esser esposto funziona in tutta generalità su qualunque catena di markov ma è stato pensato per problemi NCD e in particolare per il pagerank di google, alla fine sarà giustificato il perchè funziona meglio per questi problemi

Si utilizzeranno le notazioni introdotte fin ora, dunque noto il vettore di probabilità $\boldsymbol{\phi}$ della catena di markov con matrice di transizione $\boldsymbol{Q}$ e si vuole calcolare $\boldsymbol{\pi}$ il vettore di probabilità della catena dopo l'upadate che avrà matrice di transizione $\boldsymbol{P}$ dove si richiede l'irriducibilità (ciò è chiaramente verificato nel caso di google). Non si chiede che il numero di stati sia costante, anzi in generale m sarà il numero di stati prima dell'update ed n dopo l'update.

Algoritmo di aggregazione iterativa

(i) Partizionare l'insieme degli stati $S = G \cup \overline{G}$ e riordinare la matrice di transizione $\boldsymbol{P}$

(ii) $\overline{\boldsymbol{\phi}}^T \longleftarrow$ componenti di $\boldsymbol{\phi}^T$ relative agli stati di $\overline{G}$

(iii) $\boldsymbol{s}^T \longleftarrow \overline{\boldsymbol{\phi}}^T / \overline{\boldsymbol{\phi}}^T \boldsymbol{e}$ (approssimazione iniziale dell vettore di probabilità della catena censurata)

(iv) Iterare fino a convergenza

(1) $\boldsymbol{A} \longleftarrow \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} \boldsymbol{e} \\ \boldsymbol{s}^T \boldsymbol{P}_{2,1} & 1 - \boldsymbol{s}^T \boldsymbol{P}_{2,1} \boldsymbol{e} \end{pmatrix}$

(2) $\boldsymbol{\alpha}^T \longleftarrow (\alpha_1, \ldots, \alpha_{g+1})$ (vettore di probabilità di $\boldsymbol{A}$)

(3) $\boldsymbol{\chi}^T \longleftarrow (\alpha_1, \ldots, \alpha_g | \alpha_{g+1} \boldsymbol{s}^T)$

(4) $\boldsymbol{\psi}^T \longleftarrow \boldsymbol{\chi}^T \boldsymbol{P}$

(5) Se $\|\boldsymbol{\psi} - \boldsymbol{\chi}\| < \tau$ per una data tolleranza τ allora l'algoritmo termina, altrimenti $\boldsymbol{s}^T \longleftarrow \overline{\boldsymbol{\psi}}^T/(\overline{\boldsymbol{\psi}}^T \cdot \boldsymbol{e})$ e si ritorna al punto (1).

Per dimostrare la convergenza dell'algoritmo di aggregazione iterata sono necessari due lemmi

Richiamo 2.1. Data una M-matrice singolare e irriducibile (oppure non singolare), tutte le sue sottomatrici non banali sono M-matrici non singolari e le loro inverse sono a coefficienti positivi. In particolare $\boldsymbol{I} - \boldsymbol{P}$ è una M-matrice singolare.

Lemma 2.1. Il complemento di Schur di una matrice stocastica irriducibile è una matrice stocastica

Dimostrazione. Si consideri $\boldsymbol{P}$ una matrice stocastica irriducibile partizionata a blocchi

$$\boldsymbol{P} = \left(\begin{array}{c|c} \boldsymbol{P}_1 & \boldsymbol{P}_2 \\ \hline \boldsymbol{P}_3 & \boldsymbol{P}_4 \end{array}\right)$$

Dunque il complemento di Schur sarà $\widehat{\boldsymbol{P}} = \boldsymbol{P}_4 + \boldsymbol{P}_3(\boldsymbol{I} - \boldsymbol{P}_1)^{-1}\boldsymbol{P}_2$, per ipotesi $\boldsymbol{P}$ è stocastica, quindi $(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{e} = \boldsymbol{0}$ che scritta a blocchi diventa

$$\begin{cases} (\boldsymbol{I} - \boldsymbol{P}_1)\boldsymbol{e} + \boldsymbol{P}_2\boldsymbol{e} = \boldsymbol{0} \\ \boldsymbol{P}_3\boldsymbol{e} + (\boldsymbol{I} - \boldsymbol{P}_4)\boldsymbol{e} = \boldsymbol{0} \end{cases}$$

dalla prima equazione si ottiene

$$\boldsymbol{e} = -(\boldsymbol{I} - \boldsymbol{P}_1)^{-1}\boldsymbol{P}_2\boldsymbol{e}$$

che sostituita nella seconda da

$$-\boldsymbol{P}_3(\boldsymbol{I} - \boldsymbol{P}_1)^{-1}\boldsymbol{P}_2\boldsymbol{e} - \boldsymbol{P}_4\boldsymbol{e} + \boldsymbol{e} = \boldsymbol{0}$$

ovvero $\widehat{\boldsymbol{P}}\boldsymbol{e} - \boldsymbol{e} = \boldsymbol{0}$ □

Lemma 2.2. Sia $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ la matrice di transizione di una catena di markov irriducibile, si consideri il partizionamento

$$A = \left(\begin{array}{c|c} \boldsymbol{B} & \boldsymbol{\gamma} \\ \hline \boldsymbol{\alpha}^T & c \end{array}\right) \quad \text{con} \quad \begin{array}{l} \boldsymbol{B} \in \mathbb{R}^{(n-1)\times(n-1)} \\ \boldsymbol{\alpha}, \boldsymbol{\gamma} \in \mathbb{R}^{n-1} \\ c \in \mathbb{R} \end{array}$$

Allora il vettore di probabilità è

$$\boldsymbol{\pi}^T = \beta(\boldsymbol{\alpha}^T(\boldsymbol{I} - \boldsymbol{B})^{-1}|1)$$

dove β è la costante di normalizzazione, ovvero usando il richiamo 2.1

$$\beta = (\boldsymbol{\alpha}^T(\boldsymbol{I} - \boldsymbol{B})^{-1}\boldsymbol{e} + 1)^{-1}$$

Dimostrazione. Dal lemma precedente si ha che

$$c + \boldsymbol{\alpha}^T(\boldsymbol{I} - \boldsymbol{B})\boldsymbol{\gamma} = 1$$

si è usato il fatto che l'unica matrice stocastica di ordine 1 è necessariamente il numero 1.

A questo punto la dimostrazione è un semplice calcolo, infatti bisogna verificare che

$$\boldsymbol{\pi}^T A - \boldsymbol{\pi}^T = \mathbf{0}$$

equivalentemente (per alleggerire la notazione) si può mostrare che

$$(\beta^{-1}\boldsymbol{\pi})^T A - (\beta^{-1}\boldsymbol{\pi})^T = \mathbf{0}$$

$$\begin{aligned}
&(\beta^{-1}\boldsymbol{\pi})^T A - (\beta^{-1}\boldsymbol{\pi})^T && = \\
&(\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}|1)\left(\begin{array}{c|c} \boldsymbol{B} & \boldsymbol{\gamma} \\ \hline \boldsymbol{\alpha}^T & c \end{array}\right) - (\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}|1) && = \\
&(\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{B} + \boldsymbol{\alpha}^T|\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{\gamma} + c) - (\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}|1) && = \\
&(\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{B} + \boldsymbol{\alpha}^T - \boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}|\boldsymbol{\alpha}^T(\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{\gamma} + c - 1) && = \\
&(\boldsymbol{\alpha}^T[(\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{B} + \boldsymbol{I} - (\boldsymbol{I}-\boldsymbol{B})^{-1}]|0) && = \\
&(\boldsymbol{\alpha}^T[(\boldsymbol{I}-\boldsymbol{B})^{-1}(\boldsymbol{B}-\boldsymbol{I}) + \boldsymbol{I}]|0) && = \\
&(\boldsymbol{\alpha}^T[-(\boldsymbol{I}-\boldsymbol{B})^{-1}(\boldsymbol{I}-\boldsymbol{B}) + \boldsymbol{I}]|0) && = \\
&(\mathbf{0}|0) && = \\
&\mathbf{0}
\end{aligned}$$

□

Teorema 2.2. L'algoritmo di aggregazione iterativa restituisce il vettore di probabilità $\boldsymbol{\pi}$ di $\boldsymbol{P}$ per ogni scelta della partizione $S = G \cup \overline{G}$. La velocità di convergenza dell'algoritmo è esattamente la velocità con la quale $\boldsymbol{S}^n$ converge e quest'ultima è determinata dalla grandezza di $\lambda_2(S)$ (secondo autovalore dominante).

Dimostrazione. Sia $\boldsymbol{s}(0)$ l'approssimazione iniziale del vettore di probabilità della catena censurata e siano $\boldsymbol{A}(n)$, $\boldsymbol{\alpha}^T(n)$, $\boldsymbol{\chi}^T(n)$, $\boldsymbol{\phi}^T(n)$ e $\boldsymbol{s}^T(n)$ i rispettivi risultati all'iterazione i-esima.

Pertanto si avrà (usando il lemma precedente)

$$\boldsymbol{A}(n+1) = \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2}\boldsymbol{e} \\ \boldsymbol{s}^T(n)\boldsymbol{P}_{2,1} & 1 - \boldsymbol{s}(n)^T\boldsymbol{P}_{2,1}\boldsymbol{e} \end{pmatrix}$$

$$\beta_{n+1} = (1 + \boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I}-\boldsymbol{P}_{1,1})^{-1}\boldsymbol{e})^{-1}$$

$$\boldsymbol{\alpha}^T(n+1) = \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I}-\boldsymbol{P}_{1,1})^{-1}|1)$$

$$\boldsymbol{\chi}^T(n+1) = \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I}-\boldsymbol{P}_{1,1})^{-1}|\boldsymbol{s}^T(n))$$

$$\boldsymbol{\psi}^T(n+1) = \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I}-\boldsymbol{P}_{1,1})^{-1}|\boldsymbol{s}^T(n)\boldsymbol{S})$$

$$\boldsymbol{s}^T(n+1) = \boldsymbol{s}^T(n)\boldsymbol{S} = \boldsymbol{s}^T(0)\boldsymbol{S}^n$$

Dove si sottolinea che β_{n+1} è la costante di normalizzazione.

Da quanto appena scritto segue che $\boldsymbol{s}^T(n) \to \boldsymbol{s}^T$ (vettore di probabilità di $\boldsymbol{S}$) indipendentemente dal valore iniziale di $\boldsymbol{s}(0)$ e la velocità di convergenza è la stessa con cui $\boldsymbol{S}^n \to \boldsymbol{e}\boldsymbol{s}^T$.

Dato che $\boldsymbol{s}^T(n) \to \boldsymbol{s}$ si avrà

$$\beta_n \to \beta = (1 + \boldsymbol{s}^T \boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{e})^{-1}$$

Da qui si conclude che

$$\boldsymbol{\psi}^T(n) \to \beta(\boldsymbol{s}^T \boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^1 | \boldsymbol{s}^T) = \boldsymbol{\pi}^T$$

□

Nota 2.1. Ci sono un paio di passaggi che possono non esser chiari nella dimostrazione, daltronde è solo una questione di conti.

Si è detto

$$\boldsymbol{\psi}^T(n+1) = \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1} | \boldsymbol{s}^T(n)\boldsymbol{S})$$

L'algoritmo prevede che

$$\boldsymbol{\psi}^T(n+1) = \boldsymbol{\chi}(n)^T \boldsymbol{P}$$

Bisognerebbe mostrare che sono equivalenti, infatti

$$\begin{aligned}
\boldsymbol{\chi}(n)^T \boldsymbol{P} &= \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1} | \boldsymbol{s}^T(n))\boldsymbol{P} \\
&= \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1} | \boldsymbol{s}^T(n)) \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} \\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} \end{pmatrix} \\
&= \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{P}_{1,1} + \boldsymbol{s}^T(n)\boldsymbol{P}_{2,1} | \boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{P}_{1,2} + \boldsymbol{s}^T(n)\boldsymbol{P}_{2,2}) \\
&= \beta_{n+1}(\boldsymbol{s}^T(n)[\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{P}_{1,1} + \boldsymbol{P}_{2,1}] | \boldsymbol{s}^T(n)\boldsymbol{S}) \\
&= \beta_{n+1}(\boldsymbol{s}^T(n)[\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}\boldsymbol{P}_{1,1} + (\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})\boldsymbol{P}_{2,1}] | \boldsymbol{s}^T(n)\boldsymbol{S}) \\
&= \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1}[\boldsymbol{P}_{1,1} + (\boldsymbol{I} - \boldsymbol{P}_{1,1})] | \boldsymbol{s}^T(n)\boldsymbol{S}) \\
&= \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1} | \boldsymbol{s}^T(n)\boldsymbol{S}) \\
&= \boldsymbol{\psi}(n+1)^T
\end{aligned}$$

L'altro passaggio che potrebbe esser poco chiaro è la conclusione

$$\boldsymbol{\psi}^T(n) \to \beta(\boldsymbol{s}^T \boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^1 | \boldsymbol{s}^T)$$

Bisognerebbe provare che questo è effettivamente il vettore di proabilità di $\boldsymbol{P}$ daltronde basta ricordare che il vettore di probabilità della catena madre è

$$\boldsymbol{\alpha}^T = \beta_{n+1}(\boldsymbol{s}^T(n)\boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^{-1} | 1)$$

Quindi usando (5) segue che

$$\boldsymbol{\pi}^T = \beta(\boldsymbol{s}^T \boldsymbol{P}_{2,1}(\boldsymbol{I} - \boldsymbol{P}_{1,1})^1 | \boldsymbol{s}^T)$$

Conclusioni

In definitiva si ha che l'algoritmo di aggregazione iterata non chiede più passi di quanti ne richiederebbe il metodo delle potenze ma ogni passo è computazionalmente più costoso, inoltre si osserva che la velocità di convergenza dipende dal secondo autovalore dominante di $\boldsymbol{S}$, non è difficile intuire che per $|G| = g$ grande (quando la dimensione della catena censurata cresce) si avrà convergenza in meno passi ma chiaramente ogni passo richiederà più tempo. Pertanto bisognerà scegliere una partizione iniziale $G \cup \overline{G}$ opportuna. Per tale scelta di possono seguire diverse strade quasi tutte basate su euristiche legate al problema particolare di google e in generale alle reti di tipo small-world, ovvero tali euristiche sfruttano la power-law, la scale-free , la presenza di cluster e chiaramente la sparsità. Per parlare di tali euristiche è necessario introdurre dunque il concetto di G-set e L-shape.

2.2.4 Scelta della parizione

L'obbiettivo è quello di fornire delle euristiche per la scelta della partizione degli stati $S = G \cup \overline{G}$.

Una prima idea per la scelta di tale partizione potrebbe esser quella di includere in G tutti gli stati in qualche senso "vicini" agli aggiornamenti, per misurare tale vicinanza si possono utilizzare ad esempio nozioni di teoria dei grafi quale il *flusso transiente*, ovvero osservare la magnitudine di $\boldsymbol{x}_{j+1}^T = \boldsymbol{x}_j^T \boldsymbol{P}$ dove si faranno poche iterazioni.

Un'euristica decisamente più efficace è quella basata sulla proprietà scale-free e power-low del web, pertanto saranno rapidamente presentati questi due concetti.

Si considerino le funzioni seguenti

$$P(d) = |\{\text{pagine web che hanno } d \text{ link verso altre pagine web}\}|$$
$$Q(d) = |\{\text{pagine web che che sono linkate da altre } d \text{ pagine web}\}|$$

Tali funzioni sono note rispettivamente come distribuzione del'*in-degree* e dell'*out-degree*.

Nella rete web è stato osservato che tali funzioni sono di tipo esponenziale (power-low) ovvero

$$P(d) \propto \frac{1}{d^k}$$
$$Q(d) \propto \frac{1}{d^h}$$

Inoltre quando la dimensione della rete cambia (ad esempio dopo un update) queste funzioni non cambiano (scale-free), pertanto si hanno degli invarianti (seppur euristici) per update. In particolare $k \approx 2.7$ e $h \approx 2.1$.

D'ora in avanti si considererà solo l' *in-degree*, dato che questo rispetta la power-low, si trova che il vettore di probabilità (ordinato in modo decrescente rispetto l'*in-degree*) rispetterà anche lui una power-low, in questo contesto si parla di L-shape. L'euristica che si segue è quella di mettere in G i nuovi stati e i primi g-stati in ordine di grandezza in base all'*in-degree*. Queste particolari partizioni sono note come G-sets.

Per stabilire quanto grade deve essere g bisogna pensare ad altre euristiche o a delle condizioni particolari dipendenti dal problema. Per i dettagli sperimentali si consulti [3].

3 Ordinal ranking per il pagerank di Google

Tutto quello che riguarda l'ordinal ranking è preso da [6]

3.1 Descrizione del problema

Si è descritto il modello di Google e come effettuare l'update del pagerank, daltronde ci sono molti aspetti che non sono stati esaminati, uno dei più importanti è la computazione stessa del pagerank. Brevemente si ricorda che il metodo che si utilizza per la computazione del pagerank è il metodo delle potenze

$$\boldsymbol{x}^{(n+1)T} = \boldsymbol{x}^{(n)T}\boldsymbol{A}$$

Daltronde non è chiaro quale condizione di arresto scegliere, in genere per quasi tutte le applicazioni si chiede che il residuo sia abbastanza piccolo ovvero

$$\|\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}\| < \varepsilon$$

Daltronde sorgono spontane le seguenti domande

- Quale norma scegliere?
- Quale ε garantisce una buona approssimazione?
- Per k grande, $x^{(k)}(i) > x^{(k)}(j)$ implica che $\pi(i) > \pi(j)$?

Innanzitutto si può mostrare che non è efficiente (nel caso del pagerank) scegliere una qualsiasi norma geometrica (come le norme $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$), è necessario introdurre una nuova distanza, nota come *rank distance* e non è sufficiente chiedere che il residuo sia abbastanza piccolo, lo si vedrà bene nella sottosezione successiva.

3.2 Distanza indotta dall'ordinal ranking

Definizione 3.1 (Ordinal rank)**.** Sia $\boldsymbol{x} = (x_i)_{i=1,\dots,n}$ e sia σ la permutazione che ordina gli elementi di $\boldsymbol{x}$ in ordine decrescente, ovvero

$$x(\sigma(1)) \geq x(\sigma(2)) \geq \dots x(\sigma(n))$$

allora si definisce *ordinal rank* di un elemento come $\text{Orank}(x(i)) = \sigma(i)$ mentre l' *ordinal rank* di tutto il vettore sarà

$$\text{Orank}(\boldsymbol{x}) = (\text{Orank}(x(1)), \dots, \text{Orank}(x(n)))^T$$

Definizione 3.2 (Distanza indotta dall'ordinal rank)**.** Dati due vettori $\boldsymbol{x}$ e $\boldsymbol{y}$, ciascuno con elementi distinti, si definisce la *rank distance*

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{1 \leq i,j \leq n} \delta_{x,y}(i,j)$$

dove

$$\delta_{x,y}(i,j) = \begin{cases} 1 & \text{se} \quad \text{Orank}(x_i) < \text{Orank}(x_j) \quad \text{e} \quad \text{Orank}(y_i) > \text{Orank}(y_j) \\ 0 & \text{altrimenti} \end{cases}$$

Come già annunciato, il fatto che sia piccola la distanza geometrica o la *rank distance* tra due successive iterate (il residuo) non assicura un corretto ranking, si considererà invece la *rank distance* tra l'iterata $\boldsymbol{x}^{(n)}$ e il pagerank $\boldsymbol{\pi}$, si otterranno tali informazioni su questa distanza dalla distanza geometrica tra due successive iterate e dalla particolare struttura della matrice di google, il tutto segue dai seguenti teoremi.

L'obbiettivo che si vuole raggiungere è la determinazione dei primi k elementi del pagerank.

Teorema 3.1. Sia $\boldsymbol{x} \geq 0$ con $\|\boldsymbol{x}\|_1 = 1$ un'approssimazione di $\boldsymbol{\pi}$ e $\beta \geq \|\boldsymbol{x} - \boldsymbol{\pi}\|_1$. Se $x(i) > x(j) + \beta$ allora $\pi(i) > \pi(j)$

Notazione 3.1. Dato $\boldsymbol{x}$, sia $\boldsymbol{Q}$ una matrice di permutazione tale che mette in ordine decrescente le componenti di $\boldsymbol{x}$

$$\tilde{\boldsymbol{x}} := \boldsymbol{Q}\boldsymbol{x} = (\tilde{x}(1), \ldots, \tilde{x}(n))^T$$

Si definisca inoltre

$$\tilde{\boldsymbol{\pi}} := \boldsymbol{Q}\boldsymbol{\pi} = (\tilde{\pi}(1), \ldots, \tilde{\pi}(n))$$

non è detto che le componenti di $\tilde{\boldsymbol{\pi}}$ siano in ordine decrescente.

Lemma 3.1. Se $\tilde{x}(k) > \tilde{x}(k+1) + \beta$ allora $\text{Orank}(\tilde{\pi}(i)) \leq k$ per $1 \leq i \leq k$ e $\text{Orank}(\tilde{\pi}(j)) \geq k+1$ per $k+1 \leq j \leq n$.

Lemma 3.2 (rank esatto per le prime k componenti)**.** Se $\tilde{x}(k) > \tilde{x}(k+1) + \beta$ e $\tilde{x}(k+1) > \tilde{x}(k+2) + \beta$ allora $\text{Orank}(\tilde{\pi}(k+1)) = k+1$

Conclusione Se $\boldsymbol{x}$ è tale che $\|\boldsymbol{x} - \boldsymbol{\pi}\|_1 \leq \beta$ e se $\tilde{x}(k) > \tilde{x}(k+1) + \beta$ e $\tilde{x}(k+1) > \tilde{x}(k+2) + \beta$ allora le prime k pagine web con rank più alto sono le prime k componenti di $\text{Orank}(\boldsymbol{x})$.

Chiaramente resta ancora la questione del come calcolare β, fatto questo si potranno calcolare le prime k pagine web con rank più alto come mostrato in questa sottosezione.

3.3 Limitazione dell'errore nel metodo delle potenze

Di seguito sono presentati i metodi principali utilizzati per il calcolo di β, si evita di dimostrare tutte le disuguaglianze che si basano su conti, ad ogni modo i dettagli si trovano in [6]

- Simple bound

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{\pi}\|_1 \leq \gamma^k \|_1 \boldsymbol{x}^{(0)} - \boldsymbol{\pi}\|_1 \leq 2\gamma^k$$

- Backward-looking bounds

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{\pi}\|_1 \leq \frac{\gamma^j}{1-\gamma^j} \|\boldsymbol{x}^{(k-j)} - \boldsymbol{x}^{(k)}\|_1 \qquad 1 \leq j \leq k$$

- Forward-looking bounds

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{\pi}\|_1 \leq \frac{\|\boldsymbol{x}^{(k+j)} - \boldsymbol{x}^{(k)}\|_1}{1-\gamma^j}$$

- Two-level, forward-looking bounds

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{\pi}\|_1 \leq \|\boldsymbol{x}^{(k+j)} - \boldsymbol{x}^{(k)}\|_1 + \frac{\|\boldsymbol{x}^{(k+j+i)} - x^{(k+j)}\|_1}{1-\gamma^i} \qquad j, i \geq 1$$

Dove si ricorda che γ è la costante utilizzata per fare la combinazione convessa tra la matrice stocastica di google e la correzione di rango 1.

4 Algoritmo EigenTrust per le reti P2P

Segue una breve ed incompleta descrizione dell'algoritmo dell'eigentrust per mostrare un altro modello matematico simile a quello di google, tutto quello che riguarda questo argomento lo si trova in [5].

4.1 Descrizione del problema

Le reti p2p (peer to peer) sono utilizzate per lo scambio di files tra utenti. Un utente che fa parte di una di queste reti mette a disposizione alcuni files e gli altri utenti possono scaricarli da lui e viceversa. I sistemi p2p più famosi sono eMule, *Napster, torrent* .... Il problema di questi sistemi è la presenza di utenti nocivi, ovvero ci sono utenti che condividono files che in realtà sono virus, spyware o semplicemente files non corretti. Non è difficile diffondere virus in questo modo, è sufficiente che un utente nocivo dia al suo file un nome accattivante e cambi formato. Ad esempio potrebbe chiamarlo *"ubuntu11.10.rar"* in modo che un utente che vuole quella versione di ubuntu scarichi questo file senza sapere che in realtà si sta danneggiando. Non ci sono "leggi" che che puniscono gli utenti nocivi (a meno che questi non diffondano materiale illegale) quindi è necessario che la rete da se riesca ad inviduare questi ultimi ed eliminarli, ci sono vari metodi per farlo e c'è stata un'evoluzione di questi, il metodo che attualmente viene utilizzato è quello dell'*eigentrust* che è descritto nella sottosezione successiva.

4.2 EigenTrust

Si consideri $S = \{i_1, \ldots, i_n\}$ l'insieme degli utenti di una rete p2p, l'obbiettivo è attribuire un valore di fiducia ad ogni utente, per farlo si inizia con il definire il *valore di fiducia locale*. Si supponga che l'utente i abbia scaricato un file dall'utente j, allora se è soddisfatto della transizione attribuirà all'utente j un punto di fiducia in caso contrario glielo toglierà. In definitiva l'utente i attribuirà all'utente j un numero

$$s_{i,j} = \text{sat}(i,j) - \text{unsat}(i,j)$$

Ovvero $s_{i,j}$ è il numero di download (che l'utente i ha effettuato dall'utente j) andati a buon fine meno quelli andati male (files non corretti, download interrotto, ...).

Il problema è che $s_{i,j}$ può esser un numero negativo, daltronde se è un numero negativo sicuramente i ha una "pessima opinione" di j quindi di certo non scaricherà altri files da lui, pertanto è possibile evitare che le opinioni siano negative e dire che se $s_{i,j} = 0$ allora i preferisce (se possibile) scaricare il file che ha cercato da qualun altro e non da j.

Pertanto si definisce il *valore di fiducia locale normalizzato*

$$c_{i,j} = \frac{\max(s_{i,j}, 0)}{\sum_{j=1}^{n} \max(s_{i,j}, 0)}$$

Si osserva che se il denominatore è nullo allora l'utente i non ha eseguito alcun download (oppure ha avuto da ogni utente tanti successi quanti insuccessi), in tal caso i non ha un'opinione di j, quindi si dovrà attribuire un *valore di pregiudizio*, ma se ne parlerà a breve, per ora si supponga che $c_{i,j}$ sia sempre definito (ovvero ogni utente ha ottenuto almeno una transizione terminata con successo). Si denoti con $\boldsymbol{C} = (c_{i,j})$ la matrice con contiene i valori di fiducia. Si osserva subito che questa matrice è stocastica.

Tutto quello che è stato detto sin ora era noto anche molti anni fa, l'idea innovativa che hanno avuto gli autori di [5] è quella di considerare transitiva la relazione di fiducia, ovvero se l'utente i si fida dell'utente j e se l'utente j si fida dell'utente k allora anche i si fiderà di k (pur non avendoci mai avuto a che fare) e peserà la sua fiducia con l'opinione che ha di j.
Quindi l'utente i chiederà ai suoi conoscenti (utenti da cui ha scaricato files) che opinione hanno degli altri utenti che a loro volta conoscono.

$$t_{i,k} = \sum_{j=1}^{n} c_{i,j} c_{j,k}$$

Quindi $t_{i,k}$ è l'opinione che i ha di k basandosi sulla sua eventuale esperienza e sull'opinione che i suoi amici hanno di k. In generale si avrà

$$\boldsymbol{t}_i = \boldsymbol{C}^T \boldsymbol{c}_i$$

Dove $\boldsymbol{t}_i = (t_{i,1}, \ldots, t_{i,n})^T$ e $\boldsymbol{c}_i = (c_{i,1}, \ldots, c_{i,n})^T$. Daltronde l'utente i potreppe chiedere le opinioni agli amici dei suoi amici e quindi con la stessa costruzione

$$\boldsymbol{t}_i^{(2)} = (\boldsymbol{C}^T)^2 \boldsymbol{c}_i$$

Può continuare così a chiedere l'opinione degli amici degli amici $\ldots n$ volte, quindi ottenere

$$\boldsymbol{t}_i^{(n)} = (\boldsymbol{C}^T)^n \boldsymbol{c}_i$$

Se $\boldsymbol{C}$ è irriducibile e aperiodica allora $\boldsymbol{t}_i^{(n)}$ converge ad un vettore $\boldsymbol{t}_i$, non è difficile mostrare che il vettore $\boldsymbol{t}_i$ è uguale per ogni utente i di partenza, quindi in generale $\boldsymbol{t}_i^{(n)} \to \boldsymbol{t}$, per la teoria esposta fin ora è cosa nota che $\boldsymbol{t}$ è l'autovettore sinistro dominante di $\boldsymbol{C}$.

4.3 Interpretazione probabilistica

Non è difficile trovare l'interpretazione probabilistica con le catene di Markov, infatti è possibile pensare ad una sorta di random walk come nel caso di google, ovvero l'utente i scaricherà dall'utente j con probabilità $c_{i,j}$, a sua volta l'utente j scaricherà dall'utente k con probabilità $c_{j,k}$ .... Quindi ci si chiede dopo n passaggi, partendo da i, a quale utente si è arrivati? Questo equivale a chiedersi qual'è il vettore di probabilità della catena di markov che ha come stati S l'insieme degli utenti e come ha matrice di transizione $\boldsymbol{C}$.

4.4 Pregiudizio in una rete p2p

In questo contesto non è il caso di approfondire troppo la questione del pregiudizio che è decisamente più complessa di come sarà esposta in questa sottosezione, è necessario parlarne solo per introdurre una condizione di irriducibilità della catena di Markov.

Supponiamo che alcuni utenti creino una rete p2p, allora è sensato supporre che almeno loro non siano utenti nocivi dato che a priori non avrebbero alcun interesse nel distruggere la rete che loro stessi hanno

creato, quindi sia P l'insieme di questi utenti, detti anche *utenti pregiudicati*, si definisca la *distribuzione di pregiudizio* come $\boldsymbol{p} = (p_i)$ con

$$p_i = \begin{cases} \dfrac{1}{|P|} & \text{se} \quad i \in P \\ 0 & \text{altrimenti} \end{cases}$$

Ora è possibile correggere il modello ridefinendo il *valore di fiducia locale normalizzato* come

$$c_{i,j} = \begin{cases} \dfrac{\max(s_{i,j}, 0)}{\displaystyle\sum_{j=1}^{n} \max(s_{i,j}, 0)} & \text{se} \qquad \displaystyle\sum_{j=1}^{n} \max(s_{i,j}, 0) \neq 0 \\ p_j & \text{altrimenti} \end{cases}$$

L'idea è che un utente che è appena entrato nella rete deve pur fidarsi di qualcuno, ad esempio degli utenti pregiudicati, o in ogni caso almeno di loro deve fidarsi.

Nota 4.1. Si può mostrare che in presenza di utenti nocivi $\boldsymbol{p}^T \cdot \boldsymbol{C}^n$ converge più velocemente di $\boldsymbol{v}^T \cdot \boldsymbol{C}^n$.

In questo modo il problema è ben posto, si ha l'irriducibilità e quindi la soluzione esiste, quindi ci sarà un vettore $\boldsymbol{t}$ di fiducia tale che t_i sarà il *valore globale di fiducia* dell'utente i.

Resta un'ultima considerazione: gli utenti nocivi potrebbero collaborare tra loro e attribuirsi un valore di fiducia alto per rendere inaffidabile la computazione di $\boldsymbol{t}$. Si consideri $\boldsymbol{C}$ come descritta all'inizio, senza la modifica con i pregiudizi, allora l'idea è quella di considerare come matrice di transizione

$$\boldsymbol{A} = \gamma \boldsymbol{C} + (1 - \gamma) \boldsymbol{e} \cdot \boldsymbol{p}^T$$

Dove γ è una costante assegnata (come nel modello di google) e questo equivale a dire che l'utente i attribuisce utomaticamente ad ogni utente pregiudicato almeno una certa fiducia. Pensando a ramdom walk questo equivale a dare una via di fuga da un cluster di utenti nocivi introducendo la possibilità di spostarsi su un altro utente a caso.

4.5 Conclusioni

Volontariamente non si sono dati troppo dettagli sull'*EigenTrust*, l'obbiettivo era quello di presentare il modello e sottolineare le analogie con quello di google. Daltronde osserviamo che i punti che non sono stati analizzati sono:

- scelta degli utenti pregiudicati
- scelta della costante γ
- come implicitamente un utente computa i *valori di fiducia locali*

L'ultimo punto è solo una questione tecnica, per i precedenti è necessario dilungarsi troppo, daltronde tutti i dettagli si trovano in [5] e in numerosi altri lavori sull' *EigenTrust*, quindi si conclude qui questa breve descrizione.

Riferimenti bibliografici

[1] Dario Bini, http://it.mathacademy.ws/il-modello-del-pagerank-di-google/

[2] Amy N. Langville and Carl D. Meyer, Google's PageRank and Beyond:The Science of Search Engine Rankings, 2006

[3] Amy N. Langville and Carl D. Meyer, Updating Markov Chains with an eye on Google's Pagerank, Society for Industrial and Applied Mathematics, 2006

[4] C. D. Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, Philadelphia, 2000 (disponibile online su http://linearalgebra.net/DownloadChapters.html)

[5] Sepandar D. Kamvar, Mario T. Schlosser , Hector Garcia-Molina, The EigenTrust Algorithm for Reputation Management in P2P Networks, Stanford University, 2003

[6] Rebecca S. Wills, Ilse C.F. Ipsen, Ordinal Ranking for Google Pagerank, SIAM J. Matrix Anal. Appl.Vol. 30, No. 4, pp. 1677–1696, Society for Industrial and Applied Mathematics, 2009