EQUAZIONI LINEARI E MATRICI: LA MATEMATICA IN RETE

DIMITRI BREDA, CLARA VERONESE

3.1 Introduzione

Queste pagine riassumono attività svolte per il Laboratorio "Equazioni lineari e matrici: la matematica in rete" del Piano Nazionale Lauree Scientifiche (PLS). Tali attività, rivolte a una selezione di alunni di quarta provenienti dall'Istituto Tecnico Commerciale "Antonio Zanon" di Udine, sono state progettate congiuntamente dagli autori e dall'insegnante Stefania Pividori del suddetto Istituto.

In accordo con le direttive del PLS, gli argomenti sviluppati non sono stati trattati in modo meramente nozionistico, ma con l'intento di stimolare la curiosità e l'interesse dei partecipanti nei confronti di tematiche che richiedono nozioni avanzate rispetto al programma ministeriale di matematica previsto per l'istruzione secondaria ma che, tutto sommato, si possono spesso incontrare nella vita quotidiana, soprattutto in una realtà permeata dall'intenso utilizzo della tecnologia e dell'informatica come quella attuale.

A tale scopo, vista la disponibilità dei laboratori della Facoltà di Scienze MFN dell'Università di Udine, dove si è svolta parte dell'attività, ci si è avvalsi del software MATLAB© come veicolo di apprendimento. Rimandiamo una volta per tutte gli eventuali lettori dall'interesse "insaziabile" a [80, 79], sottolineando che parte del materiale presentato è tratto da tali riferimenti, frutto del creatore stesso di Matlab, Cleve Moler.

Riguardo alla struttura delle attività, si osserva che il laboratorio è iniziato con la conferenza "Scusi prof...ma a cosa servono le equazioni?!", tenuta dal prof. Breda presso l'Istituto aderente, allo scopo di introdurre gli alunni alle tematiche trattate [22]. In seguito gli insegnanti hanno condotto una fase di "allineamento" dei requisiti, relativa ai vettori, alle matrici e ai sistemi di equazioni lineari. Per brevità escludiamo qui la trattazione di questi argomenti, rimandando alla Sezione 2 della dispensa completa [23]. Diciamo solamente che vettori e matrici sono stati introdotti nel caso bidimensionale, ovvero il più semplice possibile, tralasciandone le più profonde connotazioni algebriche e geometriche per le ragioni premesse. Al di là delle esigenze di semplificazione, questo caso si presta ottimamante all'interpretazione geometrica della matrice come trasformazione lineare nel piano cartesiano. Dopo aver introdotto - e "digerito" - operazioni quali il prodotto scalare di vettori e il prodotto righe-colonne di matrici, sono stati rivisitati sotto un'altra prospettiva i sistemi di equazioni lineari, già ben conosciuti dagli studenti. La parte finale di questa fase è stata dedicata all'analisi dei costi computazionali che comportano le operazioni base dell'algebra matriciale, accompagnando il tutto con esempi ed esercizi allo scopo di avvicinare gradualmente gli studenti all'utilizzo di Matlab.

Si è giunti quindi allo "stage" degli alunni presso i laboratori della Facoltà, dove sono state impartite le nozioni base di Matlab con relative implementazioni di codici e le fasi di approfondimento e verifica. Apprese e consolidate tali nozioni, ci siamo addentrati nel mondo della rete e del World Wide Web in particolare. L'obiettivo era quello di comprendere gli aspetti fondamentali che permettono di ordinare le informazioni in un mondo tanto vasto come il web, capire la modellizzazione matematica che è ivi nascosta, per arrivare infine a realizzare un'implementazione Matlab dell'algoritmo PAGERANK© utilizzato da GOOGLE©. Tale parte è la principale in queste brevi note.

È bene avvisare il lettore più preparato che le esigenze di semplificazione emerse nella scrittura di queste note (e di [23]) hanno spesso invaso lo spazio normalmente destinato al rigore e alla completezza caratteristici di una trattazione matematica. Sottolineiamo infine che i dati di seguito riportati si riferiscono al 2012, anno di svolgimento delle attività e di stesura di [23].

3.2 Inquadramento storico

Internet e il *World Wide Web* (di seguito semplicemente web) rappresentano fonti di problemi matematici tanto interessanti quanto difficili, almeno a livello computazionale. Tra essi potremmo annoverare la gestione dei flussi di informazioni, la loro ricerca e il loro ordinamento. E' bene sottolineare che il problema della ricerca delle informazioni nel contesto del web è abbastanza differente da ciò che avviene (ad esempio) in una biblioteca. I motivi sono diversi: la dimensione del web (numero di pagine) è di gran lunga maggiore di quella di normali cataloghi di documenti (miliardi contro milioni o migliaia); il web si evolve molto più rapidamente di quanto lo possa fare il patrimonio di una biblioteca; la struttura a *collegamenti ipertestuali* (i *link*) è una caratteristica peculiare del web che non si trova altrove. Alla luce di tali caratteristiche, l'ordinamento delle informazioni presenti nel web viene a rivestire un ruolo particolarmente cruciale. Non sorprende dunque che quest'ultimo rappresenti un argomento di ricerca attuale e tra i più gettonati. Tra l'altro è una tematica molto recente, si pensi che i primi lavori scientifici apparsi in letteratura risalgono appena alla fine degli anni '90. D'altronde la prima volta che si sono connessi due computer tra loro risale al 1969, il protocollo TCP/IP (Transmission Control Protocol/Internet Protocol) è stato introdotto negli anni '70, l'ipertesto HTTP (Hyper Text Transfer Protocol) negli anni '80 e il web ha fatto il suo ingresso ufficiale nella storia il 6 agosto 1991 (esattamente 46 anni dopo Hiroshima).

La tecnologia sviluppata da Google non è certamente l'unica a soddisfare l'esigenza di centinaia di milioni (poco più di un paio di miliardi) di utenti di trovare le informazioni desiderate in rete. Abbiamo deciso di occuparci esclusivamente di questa (in maniera semplificata), dato il suo notevole successo attuale. Altri metodi (vecchi e

nuovi) condividono comunque con Google alcuni aspetti fondamentali: sfruttano la caratteristica struttura a link del web (*teoria dei grafi*); descrivono il problema con matrici e vettori (*algebra lineare*); lo interpretano in senso probabilistico (*teoria della probabilità*) *e* lo risolvono con metodi efficienti (*analisi numerica*).

Nel loro famoso lavoro (disponibile in rete [86]), Larry Page e Sergey Brin (fondatori di Google) spiegano i concetti fondamentali con cui hanno stabilito le regole per ordinare il contenuto del web. Al tempo erano due studenti, entrambi venticinquenni, della Stanford University. Nella prossima sezione partiamo proprio dagli aspetti principali della loro teoria, riprendendo il percorso seguito durante la conferenza.

3.3 Descrizione

L'obiettivo che ci poniamo è quello di classificare (*to rank* in inglese) le pagine (*pages*) del web secondo la loro *importanza relativa* (e non assoluta: ci interessa sapere se una pagina è più importante di un'altra, ma non quanto è importante in assoluto). Da qui il nome dell'algoritmo noto come *PageRank* (brevettato dalla Stanford University). Ma che cos'è l'importanza (di seguito detta anche *pagerank*, appunto) di una pagina web? Tra le molte definizioni possibili, i criteri proposti da Page e Brin sono i seguenti:

(G1) il pagerank di una pagina è indipendente dal suo contenuto;

(G2) il pagerank di una pagina è trasferito in parti uguali alle pagine collegate in uscita da questa (mediante gli *outlink*);

(G3) il pagerank di una pagina è la somma delle frazioni di pagerank delle pagine collegate a questa in entrata (mediante gli *inlink*).

I collegamenti di cui sopra si riferiscono, ovviamente, ai link ipertestuali tra le varie pagine. Consideriamo, ad esempio, la piccolissima porzione di web schematizzata in Figura 3.1. La pagina 1 ha un pagerank 100 e, avendo 2 outlink, ne trasferirà 50 alla pagina 3 e 50 alla pagina 4 (criterio (G2)). La pagina 3 ha due inlink, riceve così un pagerank di 50 (metà) dalla pagina 1 e un pagerank di 10 (un terzo) dalla pagina 2, totalizzando quindi un pagerank di 60 (criterio (G3)). Infine, osserviamo come tali calcoli siano stati fatti senza conoscere il contenuto delle pagine (criterio (G1)).

Partendo da queste tre semplici regole svilupperemo il modello matematico su cui si basa il motore di ricerca Google. Inizieremo con un modello piuttosto primitivo, che per questo richiede alcune ipotesi semplificative rispetto alla situazione reale (Sezione 3.3). Impareremo poi a risolvere in maniera efficiente il problema matematico associato (Sezione 3.3). Infine rimuoveremo le ipotesi iniziali per avvicinarci il più possibile alla realtà (Sezione 3.3). Nel frattempo, lungo il percorso, forniremo un'interpretazione alternativa a quella algebrica.

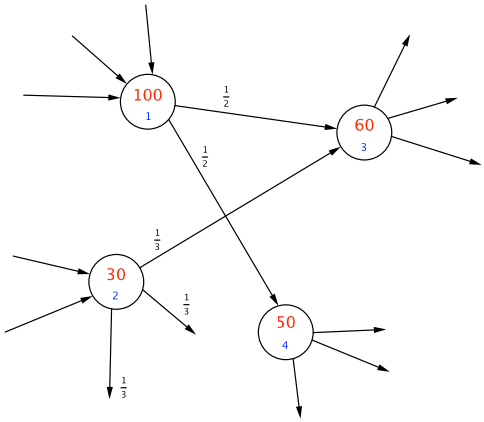

Figura 3.1: Pagine web e pagerank.

Matrici di connettività e navigatori casuali

Innanzitutto, numeriamo le pagine web da 1 ad n e indichiamo con x_i il pagerank della pagina i, $i = 1, 2, \ldots, n$. Naturalmente n è molto grande, si stima essere attualmente intorno agli 8 miliardi (http://www.worldwidewebsize.com/). Per il momento, visto che dobbiamo introdurre i concetti generali, ci avvaleremo di esempi con n molto basso, dell'ordine dell'unità (un web piccolissimo!). Supponiamo inoltre che ogni pagina abbia almeno un link uscente, ipotesi che, come vedremo in seguito, ha un ruolo fondamentale (anche se palesemente falsa!).

Costruiamo la cosiddetta *matrice di connettività* $C \in \mathbb{R}^{n\times n}$: l'elemento c_{ij} di C, quello che occupa la riga i-esima e la colonna j-esima, vale

$$c_{ij} = \begin{cases} 1 & \text{se c'è un outlink dalla pagina } j \text{ alla pagina } i \\ 0 & \text{altrimenti.} \end{cases}$$

Ad esempio, per il web di $n = 4$ pagine rappresentato con i suoi link tramite il *grafo* (semplicemente un insieme di nodi e archi orientati) di Figura 3.2, otteniamo:

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Se osserviamo, ad esempio, la pagina 3, questa riceve i link in ingresso dalle pagine 1 e 2, per cui la riga 3 della matrice C sarà $\mathbf{c}_{\mathbf{3},:} = [1, 1, 0, 0]$. Invece, se osserviamo i link in uscita da questa, daranno luogo alla colonna 3 di C, ovvero $\mathbf{c}_{:,\mathbf{3}} = [0, 1, 0, 1]^T$, dato che indirizzano alle pagine 2 e 4.

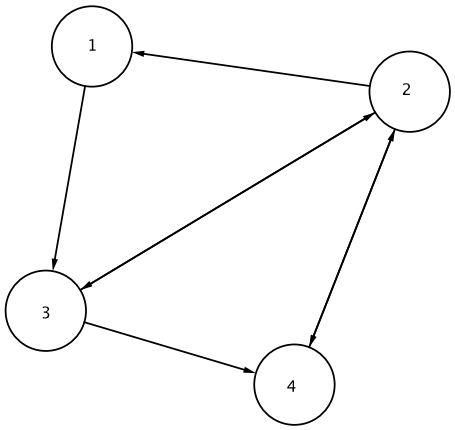

Figura 3.2: Il grafo di un web con $n = 4$ pagine.

Ora che abbiamo costruito la matrice di connettività C, effettuiamo quella che si chiama *normalizzazione*. Prendiamo dunque ciascuna colonna $\mathbf{c}_{:,j}$ di C e la dividiamo per il numero di outlink out_j della pagina relativa (possiamo farlo perchè abbiamo supposto che ve ne sia almeno uno, quindi $\text{out}_j \neq 0$). Formiamo così la nuova matrice H di colonne:

$$\mathbf{h}_{:,j} = \frac{\mathbf{c}_{:,j}}{\text{out}_j},\ j = 1, 2, \ldots, n.$$

Questa operazione realizza il corrispondente matematico del criterio (G2). Da notare che risulta essere $0 \leq h_{ij} \leq 1$ per $i, j = 1, 2, \ldots, n$ e $\sum_{i=1}^{n} h_{ij} = 1$ per $j = 1, 2, \ldots, n$. Inoltre, il numero di outlink della pagina j non è altro che la somma degli elementi di $\mathbf{c}_{:,j}$:

$$\text{out}_j = \sum_{i=1}^{n} c_{ij},\ j = 1, 2, \ldots, n.$$

Tornando all'esempio di Figura 3.2, abbiamo $\text{out}_1 = 1, \text{out}_2 = 3, \text{out}_3 = 2$ e $\text{out}_4 = 1$, da cui

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

Come stabilito dal criterio (G2) la pagina 3, ad esempio, trasferirà due metà del suo pagerank, in particolare alle pagine 2 e 4 (e così per le altre).

Come osservato, avendo assunto che ogni pagina abbia almeno un outlink, la somma degli elementi di ciascuna colonna di H vale sempre 1. Si parla in questo caso di matrice *stocastica* (a colonne). Per spiegare questa terminologia ricorriamo ad un modello interpretativo di tipo probabilistico. Immaginiamo dunque di osservare un

navigatore che sta visitando il web di Figura 3.2 e che sia obbligato a cambiare pagina ad ogni istante utilizzando *a caso* gli outlink a disposizione. Supponiamo ad esempio che ad un certo istante di tempo tale navigatore stia visitando la pagina 3 (che ha 2 outlink). Allora all'istante successivo dovrà per forza di cose trovarsi a visitare la pagina 2 o la pagina 4. Siccome si muove a caso, la probabilità di scegliere una o l'altra pagina è suddivisa in modo uguale, quindi 1/2 e 1/2 (o 50% e 50% se preferite). Per lo stesso principio, se scegliesse la pagina 4 (che ha 1 outlink), all'istante successivo si troverebbe obbligatoriamente sulla pagina 2, giungendovi con probabilità 1 (o 100%). Se invece avesse scelto la pagina 2 (che ha 3 outlink), allora all'istante successivo si ritroverebbe a visitare rispettivamente le pagine 1, 3 o 4 con uguale probabilità, ovvero 1/3, 1/3 e 1/3 (o tutte 33.333…%).

La metafora probabilistica di cui sopra va sotto il nome di *navigatore casuale*. Sotto questo punto di vista, la matrice H è una matrice di *transizione di probabilità*: il suo generico elemento h_{ij} rappresenta la probabilità che ad un certo istante il navigatore casuale passi dalla pagina j alla pagina i, seguendo ovviamente l'outlink $j \to i$. Per usare una terminologia più tecnica (e di richiamo), il navigatore casuale è un modello di *random walk* lungo il grafo associato al web esaminato e rappresenta una *catena di Markov* (Andrej [1856-1922], matematico e statistico russo).

Veniamo adesso al criterio (G3). Esso esprime una legge di tipo *ricorsivo*: il pagerank di una pagina è definito sulla base del pagerank di altre pagine (quelle che hanno outlink verso di essa in particolare). Seguendo tale ricetta, possiamo ricostruire il pagerank di ciascuna pagina. Ogni pagina, come detto, riceve solo una frazione del pagerank di ogni altra pagina che abbia un outlink diretto alla prima. Tale frazione corrisponde al relativo elemento della matrice H, ed esprime appunto la probabilità di percorrere effettivamente quel link (e quindi di trasferire tramite esso parte del pagerank della pagina di partenza). Quindi vale

$$x_i = h_{i1}x_1 + h_{i2}x_2 + \cdots + h_{in}x_n,\ i = 1, 2, \ldots, n.$$

Vista l'esperienza acquisita con l'algebra lineare, possiamo scrivere

$$\mathbf{x} = H\mathbf{x},$$

dove $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ è il cosiddetto *vettore pagerank*. Attenzione: $\mathbf{x}$ è proprio l'incognita del nostro problema!

E a che cosa corrisponde tale problema? L'espressione $\mathbf{x} = H\mathbf{x}$ non rappresenta un vero e proprio sistema lineare: il vettore incognito compare anche al posto di quello noto. D'altro canto, non è nemmeno un prodotto matrice-vettore. Se però ci ricordiamo della matrice identica I di dimensione n, allora essendo $\mathbf{x} = I\mathbf{x}$, sostituendo al primo membro otteniamo $I\mathbf{x} = H\mathbf{x}$, ovvero $(I - H)\mathbf{x} = \mathbf{0}$. Quest'ultimo è un vero e proprio sistema lineare, con matrice dei coefficienti $I - H$ $[I - A]$, vettore incognito $\mathbf{x}$ e vettore noto $\mathbf{0}$ (un caso veramente particolare). Ad esempio, sempre con riferimento al web di Figura 3.2,

otteniamo

$$\begin{bmatrix} 1 & -\frac{1}{3} & 0 & 0 \\ 0 & 1 & -\frac{1}{2} & -1 \\ -1 & -\frac{1}{3} & 1 & 0 \\ 0 & -\frac{1}{3} & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Bene, visto che siamo riusciti a tradurre il tutto in un sistema lineare, sappiamo anche come risolverlo. Purtroppo però abbiamo visto alla conferenza [22] che il metodo di eliminazione gaussiana (seguito dalla sostituzione) comporta un costo di $O(2/3n^3)$ flop. Essendo n dell'ordine dei miliardi, ci ritroveremmo a dover fare miliardi di miliardi di miliard di flop, troppi anche per i calcolatori più potenti (un milione di anni con il "K-computer"! http://www.fujitsu.com/global/about/tech/k/, http://www.top500.org/). Come superiamo questo ostacolo? La risposta nella prossima sezione.

La fortuna di Google: il metodo delle potenze e tanti zeri

Visto il costo non affrontabile del metodo di Gauss, abbandoniamo questa strada in favore di un'idea che si applica spesso in matematica: l'*iterazione*. In questo contesto, potremmo riassumere la strategia dicendo che rinunciamo alla soluzione esatta $\mathbf{x}$ ma, partendo da un'opportuna approssimazione iniziale $\mathbf{x}^{(0)}$, costruiamo una successione di soluzioni $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots$ via via sempre più vicine a quella esatta (questa almeno è la nostra speranza). Quando siamo sufficientemente "vicini", ci fermiamo. Il vantaggio consiste nel costruire queste soluzioni approssimate in modo più semplice e ad un costo computazionale molto più basso.

Tornando al problema $\mathbf{x} = H\mathbf{x}$, potremmo procedere iniziando da un certo $\mathbf{x}^{(0)}$ e calcolare $\mathbf{x}^{(1)}$ come $\mathbf{x}^{(1)} = H\mathbf{x}^{(0)}$, poi $\mathbf{x}^{(2)} = H\mathbf{x}^{(1)}$, $\mathbf{x}^{(3)} = H\mathbf{x}^{(2)}$ e via dicendo. Ad ogni passo calcoliamo dunque la soluzione "nuova" facendo il prodotto della matrice H con la soluzione "vecchia". In generale, dopo k di questi passi ci ritroviamo con

$$\mathbf{x}^{(k)} = H\mathbf{x}^{(k-1)},\ k = 1, 2, \ldots.$$

Ma quando ci si ferma? Potrebbe (il condizionale è d'obbligo per ora) essere ragionevole fermarsi quando due soluzioni successive, quindi $\mathbf{x}^{(k)}$ ed $\mathbf{x}^{(k-1)}$, differiscono di poco, cioè quando la lunghezza del vettore differenza (o errore)

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$$

è prossima allo zero, o perlomeno molto piccola. Generalizzando il Teorema di Pitagora, la lunghezza $\|\mathbf{e}^{(k)}\|$ di $\mathbf{e}^{(k)}$ risulta (in Matlab basta usare `norm`)

$$\|\mathbf{e}^{(k)}\| = \sqrt{\left(x_1^{(k)} - x_1^{(k-1)}\right)^2 + \left(x_2^{(k)} - x_2^{(k-1)}\right)^2 + \cdots + \left(x_n^{(k)} - x_n^{(k-1)}\right)^2}.$$

Ci fermeremo quindi dopo k passi, con k tale per cui $\|\mathbf{e}^{(k)}\| < \text{TOL}$ dove TOL sarà una tolleranza da noi fissata.

L'algoritmo appena presentato è noto come *metodo delle potenze* (perchè secondo voi?) e risale a circa un secolo fa [82, 83, 111]. Applichiamolo ora con riferimento al web di Figura 3.2. Questa volta ci affidiamo a Matlab. Inseriamo

```
>> H=[0,1/3,0,0;0,0,1/2,1;1,1/3,0,0;0,1/3,1/2,0]
H =
         0    0.3333         0         0
         0         0    0.5000    1.0000
    1.0000    0.3333         0         0
         0    0.3333    0.5000         0
```

Inseriamo quindi un vettore soluzione iniziale $\mathbf{x}^{(0)}$. Supponiamo ad esempio che, inizialmente, ogni pagina abbia lo stesso pagerank, quindi $1/n = 1/4$ (capiremo poi perché).

```
>> x=[1/4;1/4;1/4;1/4]
x =
    0.2500
    0.2500
    0.2500
    0.2500
```

Nella nostra implementazione Matlab chiameremo il vettore soluzione sempre `x`. Terremo conto del numero di iterazioni mediante un indice k. Siccome siamo all'inizio, diciamo che siamo al passo $k = 0$:

```
>> k=0;
```

Finalmente possiamo procedere con il metodo, che scriviamo tutto in una riga (!). Quest'ultima la ripeteremo finchè non saremo soddisfatti dell'errore (abbiamo già incontrato l'istruzione `norm`).

```
>> k=k+1,e=norm(H*x-x),x=H*x
k =
     1
e =
    0.2282
x =
    0.0833
    0.3750
    0.3333
    0.2083
>> k=k+1,e=norm(H*x-x),x=H*x
k =
     2
e =
    0.1559
x =
    0.1250
```

k	$\mathbf{e}^{(k)}$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	2.2822×10^{-1}	0.0833	0.3750	0.3333	0.2083
11	1.7455×10^{-3}	0.1247	0.3754	0.2502	0.2497
21	1.5752×10^{-5}	0.1250	0.3750	0.2500	0.2500
31	1.4602×10^{-7}	0.1250	0.3750	0.2500	0.2500
41	1.3601×10^{-9}	0.1250	0.3750	0.2500	0.2500
51	1.2677×10^{-11}	0.1250	0.3750	0.2500	0.2500
61	1.1816×10^{-13}	0.1250	0.3750	0.2500	0.2500
71	1.1012×10^{-15}	0.1250	0.3750	0.2500	0.2500

Tabella 3.1: Dati sul metodo delle potenze per il web di Figura 3.2.

```
    0.3750
    0.2083
    0.2917
>> k=k+1,e=norm(H*x-x),x=H*x
k =
     3
e =
    0.0780
x =
    0.1250
    0.3958
    0.2500
    0.2292
```

Già dopo 3 passi abbiamo ridotto l'errore. La Tabella 3.1 riporta i dati sui calcoli successivi ogni 10 passi. Osserviamo come si ottengono errori molto piccoli già con poche decine di passi. La soluzione esatta (potete verificarlo) è $\mathbf{x} = [0.125, 0.375, 0.250, 0.250]^T$: concludiamo che, supponendo 1 l'importanza di tutto il web, allora 1/8 è nella pagina 1, 3/8 nella 2 e 1/4 in ciascuna delle pagine 3 e 4.

Osservazione 3.1. La Tabella 3.1 non deve trarre in inganno: per motivi di spazio i valori di pagerank (ultime quattro colonne) sono riportati con 4 cifre significative, per cui essi possono sembrare esatti per $k \geq 21$. In realtà quelli che si vedono sono valori arrotondati a 4 cifre. Matlab li calcola effettivamente con 15 cifre significative, fatto che corrisponde alla massima precisione del sistema di rappresentazione dei numeri di macchina. Ecco perché è inutile proseguire le iterazioni dopo che l'errore scende sotto 10^{-15}: tutto quello che segue la 15^a cifra non avrebbe comunque senso!

E dopo k passi, quanto sarà costata la nostra soluzione finale $\mathbf{x}^{(k)}$ (che è comunque un'approssimazione)? Per saperlo basta capire quanto costa eseguire ogni singolo passo dell'algoritmo. Osserviamo che ogni passo comporta sempre la stessa operazione, cioè il prodotto tra la matrice di connettività normalizzata H e il vettore soluzione del passo precedente: $\mathbf{x}^{(k)} = H\mathbf{x}^{(k-1)}$, $k = 1, 2, \ldots$. Quindi un singolo passo costa in genere $O(n^2)$ flop, per un totale *di* $O(kn^2)$ flop se facciamo k

passi. Se pensiamo che in generale bastano poche decine di passi (tipicamente $k \ll n$), è già un bel risparmio. Ma non è sufficiente: siamo ancora a miliardi di miliardi di operazioni.

Abbiamo visto che ogni passo del metodo delle potenze consiste in un prodotto matrice-vettore, in particolare $H\mathbf{x}$. Abbiamo imparato che le n componenti del vettore risultante si calcolano ciascuna come prodotto scalare della relativa riga della matrice per il vettore colonna. Ad esempio per la componente i-esima:

$$x_i^{(k)} = h_{i1}x_1^{(k-1)} + h_{i2}x_2^{(k-1)} + \cdots + h_{in}x_n^{(k-1)} = \mathbf{h}_{i,:} \cdot \mathbf{x}^{(k-1)}.$$

Ma siamo sicuri che tutti gli elementi di H sono diversi da 0? È proprio qui il nocciolo della questione: $h_{ij} \neq 0$ se e solo se esiste il link $j \to i$. Ma, in generale, i link esistenti tra le pagine web sono molto meno di tutti quelli possibili. Questo si traduce nel fatto che molti elementi di H sono 0 e che quindi H ha pochi elementi diversi da 0, diciamo nnz. Di conseguenza, il costo ad ogni passo del prodotto matrice-vettore è $O(\text{nnz})$ e non più $O(n^2)$ [23, Sezione 2.5]. Questa ragione è alla base della ragguardevole riduzione del costo computazionale dato che per il web il numero di link esistenti nnz è di gran lunga inferiore al quadrato del numero delle pagine web. In altre parole, ogni pagina ha ben pochi link rispetto al totale delle pagine.

Per il web di Figura 3.2, la matrice H risulta avere $n^2 = 16$ elementi, di cui solo nnz = 7 diversi da 0 (meno della metà). 7 è ovviamente il numero totale di link esistenti, come si può verificare osservando il grafo associato. Come esempio reale, riportiamo in Figura 3.3 una rappresentazione della matrice H per il web sottostante la pagina `http://www.harvard.edu`, punto di accesso del sito della Harvard University. In tale rappresentazione, ogni punto colorato è un elemento diverso da 0 della matrice, ovvero un link. Come si evince a colpo d'occhio, i link esistenti sono davvero pochi. Infatti si contano $n = 500$ pagine, con un totale di nnz = 2636 link (notare che $n^2 = 250000$, circa 1000 volte tanto!).

Ma quanto vale nnz per il web reale? Per abbozzare una stima (ma come ordine di grandezza non andremo molto distanti), ci basiamo sul fatto che, mediamente, una pagina ha pochi link entranti. Supponiamo un centinaio. Allora se assumiamo che $n = 8$ miliardi, seguirà nnz = 800 miliardi. Quindi un passo del metodo delle potenze costa circa 800 miliardi di flop. Mediamente potremmo aspettarci che $k = 100$ passi siano più che sufficienti a raggiungere un'accuratezza soddisfacente per la soluzione. Quindi in totale dobbiamo eseguire 80 mila miliardi di flop. A questo punto decidiamo di eseguire il calcolo con un computer capace di 2 Gflops (niente di speciale oggigiorno; 1 flops = 1 flop al secondo). Ovviamente supponiamo che la memoria sia sufficiente a contenere la matrice H (e questo invece è alquanto difficile, ma questo è un altro discorso[...]). Allora per fare 80 mila miliardi di flop ad una velocità di calcolo di 2 Gflops servono 40 mila secondi, ovvero circa 11 ore: nemmeno mezza giornata. Contando che con Gauss e il "K-computer" ci volevano 1 milione di anni, possiamo essere più che soddisfatti!

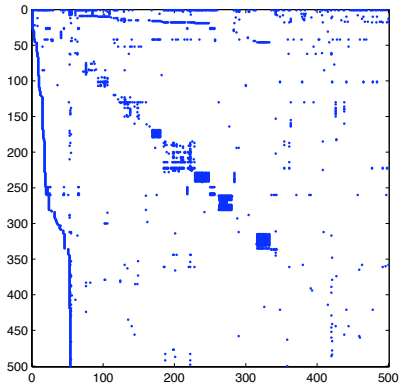

Figura 3.3: La matrice di connettività H del web di Harvard.

Pagine penzolanti e ricerche personalizzate

Vista la riduzione di costo ottenuta al termine della precedente sezione, sembrerebbe che abbiamo affrontato il problema nel modo giusto. Ma non dimentichiamo niente? Vediamo il prossimo esempio.

Consideriamo un altro web di $n = 4$ pagine rappresentato in Figura 3.4. Con le regole viste, la matrice di connettività normalizzata risulta:

$$H = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}.$$

Se ora applichiamo il metodo delle potenze si osserva che dopo ogni 3 passi ci si ritrova al punto di partenza (Tabella 3.2).

Il web dell'esempio precedente è rappresentato da una matrice H per la quale il metodo delle potenze entra in un ciclo (di *periodo* 3). Eppure soddisfa tutte le ipotesi che abbiamo assunto. Non è dunque vero che tali ipotesi sono sufficienti a garantire quella che si chiama *convergenza* del metodo, ovvero il fatto che aumentando il numero di iterazioni ci si avvicina sempre più alla soluzione esatta. Questo richiede un concetto di *limite*: $\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}$.

C'è poi un ulteriore problema: l'ipotesi che abbiamo fatto in partenza, cioè che ogni pagina abbia almeno un link in uscita ($\text{out}_j \neq 0$) non è affatto veritiera. Esistono infatti nel web reale molte pagine che non hanno link uscenti ($\text{out}_j = 0$). Anzi, sono la maggior parte. Questo si traduce in una matrice di connettività C con intere colonne vuote (cioè di soli zeri). In particolare, le pagine che non hanno link uscenti si chiamano anche *penzolanti* (in analogia con la rappresentazione del relativo grafo). Se pensiamo al modello del navigatore casuale, questo non funziona più: navigando solo tramite link, se si visita una pagina penzolante vi si rimane incastrati per sempre.

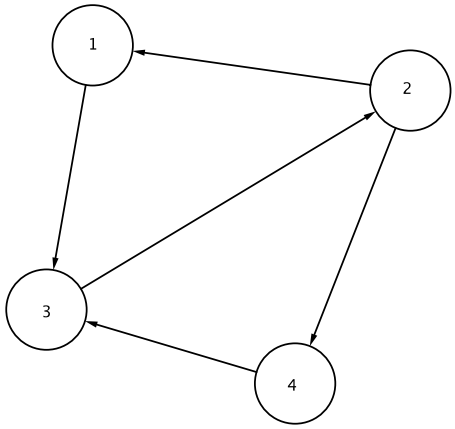

Figura 3.4: Il grafo di un altro web con $n = 4$ pagine.

k	$\mathbf{e}^{(k)}$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	0.3062	0.1250	0.2500	0.5000	0.1250
2	0.3536	0.1250	0.5000	0.2500	0.1250
3	0.3062	0.2500	0.2500	0.2500	0.2500
4	0.3062	0.1250	0.2500	0.5000	0.1250
5	0.3536	0.1250	0.5000	0.2500	0.1250
6	0.3062	0.2500	0.2500	0.2500	0.2500
7	0.3062	0.1250	0.2500	0.5000	0.1250
8	0.3536	0.1250	0.5000	0.2500	0.1250
9	0.3062	0.2500	0.2500	0.2500	0.2500
10	0.3062	0.1250	0.2500	0.5000	0.1250
11	0.3536	0.1250	0.5000	0.2500	0.1250
12	0.3062	0.2500	0.2500	0.2500	0.2500
13	0.3062	0.1250	0.2500	0.5000	0.1250
14	0.3536	0.1250	0.5000	0.2500	0.1250

Tabella 3.2: Dati sul metodo delle potenze per il web di Figura 3.4.

Per risolvere entrambi i problemi appena menzionati, gli ideatori di Google hanno scelto una "scorciatoia" che consiste nell'assicurarsi:

- da una parte di non bloccare il navigatore casuale durante la sua random walk;
- dall'altra di soddisfare ipotesi matematiche che garantiscono la convergenza del metodo delle potenze (per qualunque scelta del vettore pagerank iniziale $\mathbf{x}^{(0)}$).

Tale duplice obiettivo viene raggiunto andando a modificare la matrice originale eliminando tutti gli zeri presenti (NB: questo non è *necessario*, ma è *sufficiente*), quindi creando una matrice sempre stocastica (a colonne) ma con elementi tutti *strettamente positivi*. L'operazione viene fatta in due passi partendo dalla matrice H.

Il primo passo (da H ad H') serve ad eliminare le pagine penzolanti. Se la matrice originale H presenta delle colonne vuote, allora queste vengono riempite ovunque con $1/n$. Quindi, partendo dalla matrice C costruita come al solito, si ottiene direttamente H' come

$$h'_{ij} = \begin{cases} \dfrac{c_{ij}}{\text{out}_j} & \text{se out}_j \neq 0 \\ \dfrac{1}{n} & \text{se out}_j = 0. \end{cases}$$

Così siamo sicuri di non bloccare il navigatore casuale: se ad un dato istante questi si trova su una pagina con link uscenti, allora ne sceglie uno a caso (quindi probabilità $1/\text{out}_j$), altrimenti immette un nuovo indirizzo nella barra degli indirizzi, sempre a caso (quindi probabilità $1/n$ dato che ci sono n possibilità). Possiamo scrivere in maniera compatta la relazione precedente (o meglio il passaggio da H ad H') nel seguente modo:

$$H' = H + \frac{1}{n}\mathbf{u}\mathbf{d}^T,$$

dove $\mathbf{u} \in \mathbb{R}^{n\times 1}$ è il vettore colonna con elementi tutti 1 e $\mathbf{d} \in \mathbb{R}^{n\times 1}$ è il vettore colonna di componenti

$$d_i = \begin{cases} 1 & \text{se la pagina } i \text{ è penzolante} \\ 0 & \text{se la pagina } i \text{ non è penzolante.} \end{cases}$$

Il prodotto $\mathbf{u}\mathbf{d}^T$ segue le stesse regole del prodotto righe-colonne di matrici, perciò il prodotto del vettore colonna $\mathbf{u}$ per il vettore riga $\mathbf{d}^T$ dà luogo ad una matrice di n righe ed n colonne. In particolare, per tale matrice risultante, tutte le colonne relative a pagine non penzolanti saranno nulle, mentre quelle relative a pagine penzolanti avranno elementi tutti 1; la moltiplicazione successiva per lo scalare $1/n$ conclude l'operazione voluta, trasformando H in H'. Osservate che il prodotto $\mathbf{u}\mathbf{d}^T$ è il "contrario" del prodotto scalare, che corrisponde al prodotto righe-colonne di un vettore riga per uno colonna.

Ad esempio, per il web di Figura 3.5 (lo stesso di Figura 3.2, ma con il collegamento $1 \to 3$ rimosso), la pagina 1 è penzolante e, secondo la regola appena spiegata, risulta $\mathbf{d} = [1,0,0,0]^T$. Dunque:

$$H' = H + \frac{1}{n}\mathbf{u}\mathbf{d}^T = \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{2} & 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{3} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & 1 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 \end{bmatrix}.$$

Con Matlab:

```
>> H=[0,1/3,1/2,0;0,0,0,1;0,1/3,1/2,0;0,1/3,0,0]
H =
         0    0.3333    0.5000         0
         0         0         0    1.0000
         0    0.3333    0.5000         0
         0    0.3333         0         0
>> u=ones(n,1)
u =
     1
     1
     1
     1
```

Il vettore **d** può essere costruito nel seguente modo:

```
>> d=not(sum(H,1))'
d =
     1
     0
     0
     0
```

ovvero negando (e trasponendo) il vettore somma delle colonne di H. Proseguendo:

```
>> H1=H+u*d'/n
H1 =
    0.2500    0.3333    0.5000         0
    0.2500         0         0    1.0000
    0.2500    0.3333    0.5000         0
    0.2500    0.3333         0         0
```

Abbiamo usato anche la nuova istruzione `ones`, il cui significato dovrebbe risultare chiaro. Cosa più importante, osserviamo come Matlab esegue qualunque prodotto con "`*`", arrangiandosi a capire se sta moltiplicando scalari, vettori o matrici (e se questo si può fare!).

Il secondo passo (da H' ad H'') serve ad assicurarci la convergenza del metodo delle potenze. Per realizzarlo togliamo gli zeri rimanenti

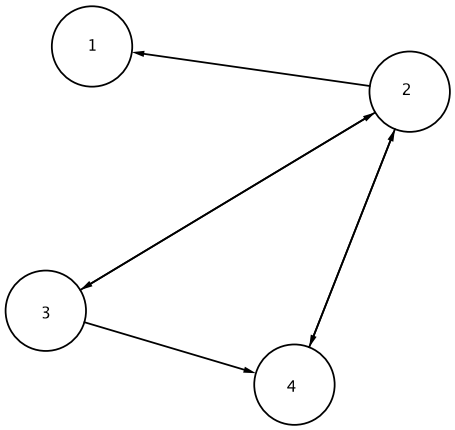

Figura 3.5: Il grafo di un web con $n = 4$ pagine, di cui una penzolante.

di H' nel seguente modo. Scegliamo un numero $\alpha \in [0,1]$, che esprime una probabilità. Imponiamo quindi al navigatore casuale

- con probabilità α di comportarsi come prima, cioè secondo H';
- con probabilità $1-\alpha$ di immettere invece un indirizzo a caso (probabilità $1/n$), indipendentemente dalla presenza o meno di link uscenti.

Certamente questo modello è più aderente alla navigazione sul web reale: infatti, le possibilità di navigazione offerte all'utente si riducono proprio a muoversi attraverso i link o a digitare nuovi indirizzi indipendentemente dalla presenza o meno di link uscenti. Matematicamente, la nuova matrice H'' si ottiene da H' come

$$h''_{ij} = \alpha h'_{ij} + (1-\alpha)\frac{1}{n}.$$

Osserviamo che in questo secondo passo stiamo semplicemente aggiungendo la quantità costante $(1-\alpha)/n$ in ogni posizione della matrice $\alpha H'$. Quindi:

$$H'' = \alpha H' + (1-\alpha)\begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \\ \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix} = \alpha H' + \frac{1-\alpha}{n}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}.$$

Ma come possiamo esprimere diversamente una matrice con tutti 1? Lo possiamo fare semplicemente con il prodotto righe-colonne del

vettore colonna $\mathbf{u}$ di tutti 1 (già introdotto in precedenza) con il suo trasposto riga $\mathbf{u}^T$, ottenendo appunto la matrice $\mathbb{R}^{n\times n}$ di tutti 1:

$$\mathbf{u}\mathbf{u}^T = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}.$$

Applichiamo quanto appena visto al passaggio da H' ad H'', ottenendo così

$$H'' = \alpha H' + (1-\alpha)\frac{\mathbf{u}\mathbf{u}^T}{n}. \tag{3.1}$$

Riassumendo, questa formula ci dice che il navigatore casuale esce da una certa pagina secondo H' con probabilità α o immettendo un nuovo indirizzo a caso con probabilità $1-\alpha$.

Ma quanto vale α? Google dichiara di utilizzare $\alpha = 0.85$. Aldilà che questo sia vero o meno, certo è che se $\alpha = 0$, allora il navigatore si muove sempre immettendo indirizzi nuovi a caso: $H'' = \mathbf{u}\mathbf{u}^T/n$, mentre se $\alpha = 1$ il navigatore si muove sempre secondo $H'' = H'$, cioè secondo i link esistenti, o immettendo un indirizzo nuovo a caso solo se la pagina in cui si trova è penzolante. Potremmo dunque affermare che una scelta di α prossimo a uno è senz'altro più fedele alla realtà. Ma la questione non finisce qui, come vedremo.

Riprendiamo la matrice H' associata al web di Figura 3.5 e supponiamo $\alpha = 0.85$. Otteniamo allora

$$H'' = 0.85\begin{bmatrix} \frac{1}{4} & \frac{1}{3} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & 1 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 \end{bmatrix} + 0.15\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \simeq \begin{bmatrix} 0.25 & 0.32 & 0.04 & 0.04 \\ 0.25 & 0.04 & 0.46 & 0.88 \\ 0.25 & 0.32 & 0.04 & 0.04 \\ 0.25 & 0.32 & 0.46 & 0.04 \end{bmatrix}.$$

Osserviamo come la matrice finale H'' rimane stocastica (a colonne). Il suo significato finale non cambia, rimanendo sempre una matrice di transizione di probabilità: ad esempio, l'elemento $h_{23} = 0.46$ (nella riga 2 e colonna 3) indica che il passaggio dalla pagina 3 alla pagina 2 avviene con il 46% di probabilità. Con Matlab:

```
>> alfa=0.85;
>> H2=alfa*H1+(1-alfa)*(u*u')/n
H2 =
    0.2500    0.3208    0.4625    0.0375
    0.2500    0.0375    0.0375    0.8875
    0.2500    0.3208    0.4625    0.0375
    0.2500    0.3208    0.0375    0.0375
```

Ora, provate voi ad eseguire il metodo delle potenze sulla matrice H''.

Nell'esempio precedente vi è stato chiesto di applicare il metodo delle potenze alla matrice H''. Osserviamo però che questa non

ha nemmeno uno zero. Quindi il costo che paghiamo è $O(kn^2)$ flop. Abbiamo dunque perso tutto quanto avevamo guadagnato prima?

La risposta è fortunatamente negativa. Basta solo ragionare e non agire di fretta. Supponiamo allora di aver già costruito H' e riconsideriamo la matrice H'' in (3.1). Assegnato $\mathbf{x}^{(0)}$, il metodo delle potenze diventa, [assegnato $\mathbf{x}^{(0)}$,] $\mathbf{x}^{(k)} = H''\mathbf{x}^{(k-1)}$, $k = 1, 2, \ldots$. Ma abbiamo visto che, essendo H'' priva di zeri, paghiamo il costo pieno. Se invece sostituiamo l'espressione di H'' nel generico passo del metodo otteniamo

$$\begin{aligned}\mathbf{x}^{(k)} &= \left(\alpha H' + (1-\alpha)\frac{\mathbf{u}\mathbf{u}^T}{n}\right)\mathbf{x}^{(k-1)} \\ &= \alpha H'\mathbf{x}^{(k-1)} + (1-\alpha)\frac{(\mathbf{u}\mathbf{u}^T)\mathbf{x}^{(k-1)}}{n} \\ &= \alpha H'\mathbf{x}^{(k-1)} + (1-\alpha)\frac{\mathbf{u}(\mathbf{u}^T\mathbf{x}^{(k-1)})}{n},\ k = 1, 2, \ldots.\end{aligned}$$

Ma quanto vale $\mathbf{u}^T\mathbf{x}^{(k-1)}$? Vediamo:

$$\mathbf{u}^T\mathbf{x}^{(k-1)} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}\begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{bmatrix} = x_1^{(k-1)} + x_2^{(k-1)} + \cdots + x_n^{(k-1)},$$

quindi questo prodotto (scalare) restituisce semplicemente la somma del pagerank di tutto il web al passo $k-1$: il pagerank totale. Per capire quanto vale, ci vuole ancora un po' di attenzione. Se $\mathbf{x}$ risolve il problema originale $\mathbf{x} = H''\mathbf{x}$, allora anche $a\mathbf{x}$ risolve lo stesso problema per qualunque numero scalare a, infatti $H''(a\mathbf{x}) = aH''\mathbf{x} = a\mathbf{x}$. Questo vuol dire che se $\mathbf{x}$ è soluzione, allora lo sono anche tutti i vettori paralleli ad $\mathbf{x}$. Dal punto di vista del nostro problema non ha importanza: se il vettore $\mathbf{x}$ ha le componenti in una certa proporzione, allora tale proporzione non cambia se prendo un vettore parallelo ad $\mathbf{x}$. Quindi l'*importanza relativa* delle pagine è sempre la stessa, ed è giusto quello che a noi interessa! Allora, se dobbiamo scegliere uno tra tutti questi vettori paralleli, prendiamo per semplicità quello le cui componenti si sommano a 1, ovvero quello per cui il pagerank totale è 1. Segue quindi $1 = x_1^{(k-1)} + x_2^{(k-1)} + \cdots + x_n^{(k-1)} = \mathbf{u}^T\mathbf{x}^{(k-1)}$. Sostituendo nel generico passo del metodo delle potenze si ottiene infine

$$\begin{aligned}\mathbf{x}^{(k)} &= \alpha H'\mathbf{x}^{(k-1)} + (1-\alpha)\frac{\mathbf{u}(\mathbf{u}^T\mathbf{x}^{(k-1)})}{n} \\ &= \alpha H'\mathbf{x}^{(k-1)} + (1-\alpha)\frac{\mathbf{u}}{n},\ k = 1, 2, \ldots.\end{aligned}$$

Ora possiamo fare un ultimo passo andando a sostituire l'espressione

esplicita di H' in funzione di H, ottenendo perciò:

$$\begin{aligned}\mathbf{x}^{(k)} &= \alpha H'\mathbf{x}^{(k-1)} + (1-\alpha)\frac{\mathbf{u}}{n} \\ &= \alpha\left(H + \frac{1}{n}\mathbf{u}\mathbf{d}^T\right)\mathbf{x}^{(k-1)} + (1-\alpha)\frac{\mathbf{u}}{n} \\ &= \alpha H\mathbf{x}^{(k-1)} + \frac{1}{n}\mathbf{u}\left[\alpha\mathbf{d}^T\mathbf{x}^{(k-1)} + (1-\alpha)\right],\ k = 1, 2, \ldots.\end{aligned}$$

La precedente formula ci dice che ciascun passo è composto da un prodotto matrice-vettore (il primo addendo) a cui sommiamo un altro vettore (il secondo addendo). La differenza è che così facendo il prodotto matrice-vettore coinvolge la matrice H, la quale ha (di nuovo) tanti zeri, ripristinando dunque il notevole risparmio computazionale che avevamo raggiunto in precedenza! Ricordiamo che il costo finale è perciò $O(k \cdot \text{nnz})$, dove nnz è il numero totale di link esistenti e k il numero di passi eseguiti con il metodo delle potenze. Per essere precisi, dovremmo tener conto anche del costo del calcolo del secondo addendo, ma facendo un po' di attenzione si può vedere che il prodotto $\mathbf{d}^T\mathbf{x}^{(k-1)}$ non fa altro che sommare l'importanza relativa delle sole pagine penzolanti al passo $k-1$, dando poi luogo ad un unico scalare corrispondente all'espressione tra parentesi quadre. Il costo massimo di questa seconda parte sarebbe dunque n, che sappiamo essere trascurabile rispetto ad nnz.

Osservazione 3.2. La scelta di avere un pagerank totale unitario si presta a completare la metafora del navigatore casuale: ciascuna componente x_i, $i = 1, 2, \ldots, n$, del vettore soluzione $\mathbf{x}$ rappresenta la probabilità di trovarsi sulla pagina i dopo un tempo di navigazione infinito. La soluzione rappresenta quello che si dice stato stazionario della catena di Markov associata e ogni sua componente, sotto questo punto di vista, è sicuramente un fedele indicatore dell'importanza della pagina considerata.

Pagerank finalmente?!

Ora siamo pronti a raccogliere tutto quanto appreso finora in un codice Matlab. Ma questo lo dovete fare voi!

Prima però dobbiamo imparare a scrivere un "programma" Matlab, quello che si chiama un *m-file function*: questo non è altro che un file di testo. Matlab ha il suo editor, altrimenti potete usare tranquillamente un notepad qualunque, a patto che salviate il file con estensione "`.m`". All'interno del corpo si scrivono le istruzioni che prevedete di eseguire, seguendo le regole di sintassi viste sinora senza alcuna differenza. È necessario aggiungere unicamente la relazione Input-Output (I/O), e tutto questo viene inserito nella prima riga di testo, detta "intestazione", che comincia **obbligatoriamente** con la parola `function`. Supponiamo quindi di dover scrivere un *m*-file che implementa un algoritmo di risoluzione di un problema che prende in Input i dati $I_1, I_2, \ldots, I_r$ e fornisce in Output i dati $O_1, O_2, \ldots, O_s$. Allora basterà scrivere come intestazione:

```
function [O1,O2,...,Os]=nome(I1,I2,...Is)
```

dove `nome` è il nome che daremo al file, che quindi sarà intitolato `nome.m`.

Ora avete tutti gli ingredienti per costruire il vostro `prank.m` partendo da una qualunque matrice di connettività non normalizzata: ad essere onesti non li avete proprio tutti, ma ci aspettiamo che facciate delle domande! Nel frattempo vediamo qualche simulazione con quello già scritto (vedi Appendice).

Esercizio 3.3. Costruite la matrice di connettività non normalizzata associata al web rappresentato in figura 3.6 e testate il vostro `prank.m` calcolando il *pagerank* con il valore di default $\alpha = 0.85$. La classifica di importanza deve risultare $P_1, P_6, P_2, P_4, P_3, P_5$.

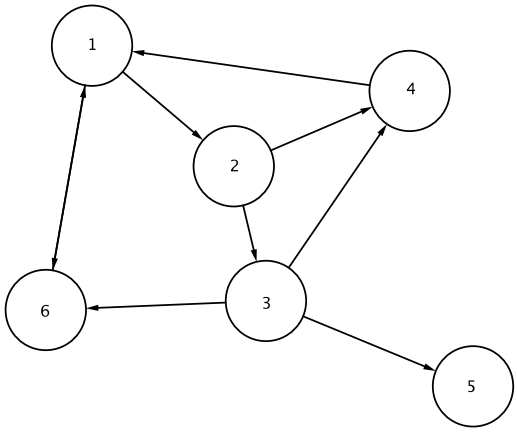

Figura 3.6: Il web di dimensione 6 per l'esercizio 3.3.

Testiamo `prank.m` sulla matrice del web di Harvard. Questa la possiamo caricare (e ispezionare) come

```
>> load harvard
>> whos
  Name          Size                Bytes  Class     Attributes
  C           500x500               15184  logical   sparse
```

Quindi la possiamo "dare in pasto" a `prank.m` come segue:

```
>> [x,ind,e,k]=prank(C,0.85,1e-5,100);
```

ottenendo l'intero vettore pagerank `x` ordinato dalla pagina più importante alla meno importante (che non riportiamo, essendo molto lungo), il vettore `ind` che contiene gli indici delle pagine ordinate secondo `x` ed infine l'errore `e` ed il numero di passi fatti `k`. Possiamo ad esempio vedere chi sono le prime 10 pagine importanti, l'errore e i passi digitando

α	0.9	0.85	0.8	0.5	0.1
k	38	28	22	10	5
ind	7	7	7	7	54
	54	54	54	54	53
	53	53	53	53	15
	18	18	18	15	7
	9	9	15	18	18
	15	15	9	9	9
	10	1	1	1	10
	1	10	10	10	222
	222	222	222	222	1
	76	55	55	55	19

Tabella 3.3: Dati sul metodo delle potenze per il web di Harvard al variare di α.

```
>> ind(1:10),e,k
ans =
     7
    54
    53
    18
     9
    15
     1
    10
   222
    55
e =
  8.7680e-006
k =
    28
```

Proviamo ora a ripetere l'esperimento per diversi valori di α (sempre tra 0 e 1), ad esempio per i valori raccolti in Tabella 3.3. Che cosa notate?

Dall'esempio precedente si osserva che più α è prossimo ad 1, più passi si devono compiere per raggiungere la stessa tolleranza. Contrariamente, più α è prossimo a 0, meno passi si devono compiere. Questo fatto indica che il metodo delle potenze è tanto più veloce quanto più α è vicino a 0. Tutto ciò può essere dimostrato rigorosamente studiando la convergenza del metodo, ma dovremmo introdurre i concetti di *autovalore* e di *autovettore* di una matrice... ma per questi è meglio iscriversi all'università!

D'altro canto possiamo notare che cambiando α cambia pure la classifica delle pagine. Quindi la questione è alquanto delicata: da una parte vorremmo essere veloci (α prossimo a 0), dall'altra vorremmo restare fedeli al web reale (α prossimo a 1).

Affrontiamo infine un'ultima faccenda curiosa. Ricordiamo cha abbiamo modificato la matrice da H ad H'' imponendo alla soluzione $\mathbf{x}$ di soddisfare

$$\mathbf{x} = \alpha H\mathbf{x} + \frac{1}{n}\mathbf{u}\left[\alpha \mathbf{d}^T\mathbf{x} + (1-\alpha)\mathbf{u}^T\mathbf{x}\right]$$

ovvero

$$\mathbf{x} = \alpha H\mathbf{x} + \frac{1}{n}\mathbf{u}\left[\alpha \mathbf{d}^T\mathbf{x} + (1-\alpha)\right], \tag{3.2}$$

dato che $\mathbf{u}^T\mathbf{x} = 1$. Come abbiamo spiegato, questo equivale a scegliere tra tutti i vettori soluzione paralleli quello che fornisce un pagerank totale pari ad 1, il che è in accordo con la metafora del navigatore casuale: deve infatti muoversi con probabilità 1 ad ogni istante. Come conseguenza si può osservare che stiamo aggiungendo al comportamento di muoversi secondo i link originali H con probabilità α (primo addendo) la possibilità con probabilità α di immettere un nuovo indirizzo a caso se la pagina è penzolante (primo addendo all'interno delle parentesi quadre) oppure di immettere un indirizzo a caso comunque con probabilità $1-\alpha$ (anche se la pagina non è penzolante, secondo addendo all'interno delle parentesi quadre) . Per quanto concerne la scelta di un nuovo indirizzo a caso, questa è uniformemente distribuita rispetto a tutti le pagine, ovvero con probabilità $u_i/n = 1/n$ per ogni pagina x_i, $i = 1,\ldots,n$. È una conseguenza della presenza del vettore $\mathbf{u}/n$ davanti all'espressione tra parentesi quadre. Questo corrisponde ad una scelta "democratica" o imparziale. Ed è quella che porta ai risultati principali che si vedono in una normale ricerca con Google. Ma non è l'unica: cosa succede, infatti, se questo nuovo indirizzo non viene scelto così casualmente, ma "forzando" la scelta verso una pagina piuttosto che un'altra? Matematicamente dovremmo sostituire le precedenti probabilità con nuove probabilità p_i/n per la pagina x_i, $i = 1,\ldots,n$, con il vincolo che $\sum_{i=1}^n \frac{p_i}{n} = 1$ ovvero $\sum_{i=1}^n p_i = n$ (come del resto accadeva per il vettore $\mathbf{u}$). Ciò equivale a modificare la (3.2) in

$$\mathbf{x} = \alpha H\mathbf{x} + \frac{1}{n}\mathbf{p}\left[\alpha \mathbf{d}^T\mathbf{x} + (1-\alpha)\right],$$

dove il vettore $\mathbf{p} = [p_1, p_2, \ldots, p_n]^T$, detto *vettore di personalizzazione*, permette appunto di forzare in qualche modo il calcolo del pagerank in modo da dare più importanza a certe pagine piuttosto che ad altre, e questo a priori scegliendo opportunamente i vari pesi p_1, p_2, etc.. Questo è quello che succede per i cosiddetti "link sponsorizzati" che compaiono evidenziati in cima (o a destra) della normale lista di risultati. Provate infatti a cercare la parola "automobile" su `http://www.google.it/`. E se proprio non siete stufi, provate pure a modificare il codice `prank.m` per adattarlo a questa nuova versione!

Esercizio 3.4. Modificate il vostro `prank.m` adottando la versione con vettore di personalizzazione $\mathbf{p}$ e, con riferimento al web rappresentato in figura 3.6, provate a scegliere $\mathbf{p}$ in modo tale da far risalire la pagina meno importante in classifica.

3.4 La voce della scuola

L'attività

Nell'anno scolastico 2011/2012 l'Istituto Tecnico "Antonio Zanon" di Udine ha partecipato al Piano Lauree Scientifiche scegliendo il percorso "Equazioni lineari e matrici: la matematica in rete" per le classi 4AM e 4BM. Non era la prima esperienza PLS per i docenti della scuola: la collaborazione il dott. Dimitri Breda, ricercatore presso il Dipartimento di Matematica e Informatica dell'Università degli Studi di Udine, era già consolidata.

L'attività è iniziata con una conferenza alla quale hanno partecipato anche tre classi quinte, per un totale di 90 studenti, oltre ad un certo numero di docenti interessati. Successivamente ha coinvolto solo le classi quarte che hanno approfondito autonomamente gli argomenti introdotti dal dott. Breda. In questa fase si è discusso, messo a fuoco ed approfondito gli spunti della conferenza che avevano suscitato particolare curiosità e interesse.

Figura 3.7: conferenza all'IT "Zanon".

Dopo una breve fase di allineamento dei requisiti di carattere matematico, un gruppo di studenti ha partecipato ai workshop nei laboratori della Facoltà di Scienze Matematiche, Fisiche e Naturali dell'Università degli Studi di Udine. Durante il "mini-stage" gli studenti, apprese e consolidate alcune nuove nozioni, sono stati introdotti nel mondo della Rete e del World Wide Web, con l'obiettivo di comprendere gli aspetti fondamentali che permettono di ordinare i risultati di una ricerca di informazioni in un mondo tanto vasto come il web, di capirne la modellizzazione matematica per arrivare, infine, a realizzare un'implementazione con il software Matlab dell'algoritmo PAGERANK© utilizzato da GOOGLE©.

La rielaborazione dell'esperienza

L'attività ha reso consapevoli gli studenti che tematiche a forte contenuto matematico, anche avanzato, si possono spesso incontrare nella vita quotidiana e che alcune conoscenze basilari di matematica (quali le equazioni lineari e l'algebra delle matrici) rivestono un ruolo fondamentale in applicazioni informatiche riguardanti la rete e il web di notevole interesse ed uso. Ne è nata una sorta di riflessione "ad alta voce" che, ripercorrendo le varie tappe dell'attività di orientamento universitario, ha permesso agli studenti della 4AM di riversare l'esperienza PLS nella narrazione ipermediale "Scusi prof...ma a cosa servono le equazioni?!", vincitrice al concorso Policultura 2012.

La narrazione

Il mitico "quesito del mattone", muovendo da alcune considerazioni sull'uso (più o meno consapevole) delle equazioni nella vita quotidiana, ha condotto ad un breve viaggio fra intuito e ragionamento, lungo le vie della logica deduttiva ed induttiva, attraverso procedimenti di astrazione e generalizzazione, fino a cogliere lo stretto nesso esistente fra la matematica, da sempre considerata una disciplina esigente, rigorosa, ma troppo distante dalla vita di tutti i giorni, e la tecnologia. Il viaggio si è concluso con la riscoperta di alcune importanti applicazioni di rete che non avrebbero mai visto la luce senza l'aiuto della matematica.

Il risultato delle attività svolte è stato qualcosa di più di un semplice racconto: l'esperienza stessa è stata ripensata e rielaborata, e ogni studente ha fornito la propria interpretazione personale di un particolare aspetto del tema affrontato.

La partecipazione al concorso

Il concorso "Policultura: la tecnologia incontra la cultura" è un'iniziativa del Politecnico di Milano dedicata alle scuole per promuovere una didattica innovativa con le tecnologie dell'informazione e della comunicazione. La giuria del Concorso Policultura ha assegnato il primo premio della sezione Senior alla narrazione della classe 4AM con questa motivazione: "*La narrazione opera una scelta molto particolare: spiega come la matematica si applichi in molti aspetti della nostra vita quotidiana. Dal punto di vista didattico, è stata lasciata grande autonomia ai ragazzi, dando spazio alla originalità dei contributi ad alla creatività personale, nel rispetto delle diverse motivazioni di partenza, disponibilità ed attitudini dimostrate dagli allievi. Dal punto di vista della comunicazione, la scelta delle immagini è originale e insieme all'alternanza delle voci narranti rende la narrazione molto fruibile. I contenuti sono molto dettagliati e precisi. Il rap finale è veramente eccezionale!*".

Il parere degli studenti

"*Come classe, abbiamo partecipato ad una conferenza che il dott. Breda dell'Università di Udine ha tenuto presso l'Aula Magna del nostro Isti-*

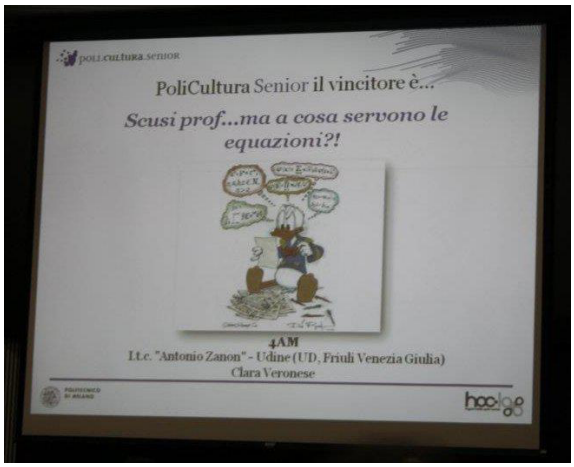

Figura 3.8: Il concorso "Policultura: la tecnologia incontra la cultura".

tuto: i contenuti avvincenti, le slide accattivanti, la presentazione originale e coinvolgente, ci hanno spinto a riflettere e a filtrare i contenuti più significativi, seguendo degli approfondimenti in classe e, per un gruppo, partecipando direttamente ad un mini-stage nei laboratori dell'Università stessa. Qui, il dott. Breda, con esempi pratici ha integrato le nostre conoscenze ed ha chiarito alcuni dubbi sulla domanda "a cosa servono le equazioni?". In quattro lezioni è riuscito, partendo dal semplice concetto di vettori, matrici e sistemi, a farci comprendere l'algoritmo che si nasconde dietro al motore di ricerca Google.

L'intenso lavoro che ha portato alla definizione di "rango di una pagina web" (pagerank), utilizzato da Page e Brin nell'ideazione del motore di ricerca più diffuso al mondo, si basa su dei calcoli riguardanti matrici di connettività e navigatori casuali. Alla fine delle lezioni abbiamo assistito ad un seminario sui social network e sul loro algoritmo di base. Quest'esperienza ci ha fatto comprendere che la matematica, anche se talvolta non amatissima da noi studenti, è un terreno fertile da cui nascono le nostre azioni quotidiane e le migliori idee di progresso.".

3.5 Conclusioni

Alla fine, un po' di numeri per rendicontare: i partecipanti alla conferenza iniziale del dott. Breda della durata di 2 ore presso lo "Zanon" erano circa una novantina (di cui la maggior parte erano studenti); una quindicina di allievi ha partecipato al mini-stage della durata totale di 7 ore presso l'Università, accompagnati da due insegnanti; un paio d'ore sono state dedicate alla co-progettazione iniziale, 4 per le fasi di allineamento dei requisiti sono condotte autonomamente dagli insegnanti e circa altrettante sono state dedicate ai lavori conseguenti il mini-stage (concorso "Policultura" e narrazione multimediale) e alle fasi finali di verifica.

Visto l'impegno complessivo e gli esiti positivi delle attività, possiamo sostanzialmente confermare quanto scritto a suo tempo alla fine dell'Introduzione in [23]: "*Con un po' di pazienza, e guidati dal motore principe della ricerca e delle innovazioni scientifiche e, in genere, culturali - la* curiosità *- la speranza alla fine di quest'esperienza è di guardare alle* matrici *e ai* sistemi *con un riguardo maggiore di quello che potremmo abitualmente concedere ad una semplice* tabella di numeri *o a delle* noiose equazioni da risolvere, *pur senza pretendere di coglierne a pieno tutte le notevoli potenzialità che celano al loro interno.*".

3.6 Ringraziamenti

Gli autori desiderano ringraziare in primis l'insegnante Stefania Pividori per la fattiva collaborazione alla progettazione e alla realizzazione delle attivià descritte in queste note, specie per quanto concerne le parti qui solamente citate, come l'allineamento dei prerequisiti, la verifica e, non ultimi, gli sviluppi condotti con i relativi studenti.

Un ringraziamento particolare va al dott. Massimo Franceschet del Dipartimento di Matematica e Informatica dell'Università degli Studi di Udine, docente di discipline informatiche ed esperto di reti sociali (e non solo), per aver fornito un contributo molto apprezzato dagli studenti attraverso il seminario "Who shall survive? Misure di centralità su reti sociali" [56].

Un ultimo ringraziamento va alla Prof.ssa Rossana Vermiglio, coordinatrice dell'intero PLS e del relativo laboratorio "La matematica c'è" in cui si è inserita l'attività qui descritta.

3.7 Codice Matlab

```
function [x,ind,e,k]=prank(C,alfa,TOL,kmax)
%PRANK pagerank.
%   [x,ind,e,k]=PRANK(C,alfa,TOL,kmax) calcola il pagerank x
% associato alla matrice di connettivita' C (sparsa) a meno
% di una tolleranza TOL e con numero massimo di iterazioni
% kmax per il metodo delle potenze. ind e' il vettore degli
% indici delle pagine web relative a C ordinato secondo il
%   pagerank x, e e' la stima dell'errore finale e k il
% numero di iterazioni eseguite.

%o verifica di sparsita'
if not(issparse(C))
    disp('ATTENZIONE: matrice C non sparsa!')
    return
end
%1: inizializzazione
n=size(C,1);%numero pagine web
u=ones(n,1);%vettore unitario
c=sum(C,1); %numero outlink
```

```
%2: costruzione matrice di connettivita' normalizzata H
H=sparse(n,n);
for j=1:n
    if c(j)~=0
        H(:,j)=C(:,j)/c(j);
    end
end
%3: costruzione vettore pagine penzolanti
d=not(sum(H,1))';
%4: metodo potenze per calcolo pagerank x
k=0;
xnew=u/n;
e=2*TOL;
while (e>=TOL)&(k<=kmax)
    k=k+1;
    xold=xnew;
    xnew=alfa*H*xold+u/n*(alfa*d'*xold+1-alfa);
    e=norm(xnew-xold);
end
%5: output ordinato secondo pagerank x
[x,ind]=sort(xnew,'descend');

%NB: i comandi tipo 'sparse' servono a non tener conto
% degli zeri nei calcoli. Ad esempio H=sparse(n,n)
% crea una matrice nxn di tutti zeri ma la memorizza
% come sparsa: quando viene modificata nelle
% successive istruzioni, gli zeri che rimangono non
% saranno comunque considerati nei calcoli.
```

Riferimenti bibliografici

[22] Dimitri Breda. «"...scusi prof...ma a cosa servono le equazioni?!"» http://www.uniud.it/didattica/facolta/scienze/la_facolta/piano-lauree-scientifiche-2010-2012/pls2/matematica-ce. 2012.

[23] Dimitri Breda, Stefania Pividori e Clara Veronese. «Equazioni lineari e matrici: la matematica in rete». http://www.uniud.it/didattica/facolta/scienze/la_facolta/piano-lauree-scientifiche-2010-2012/pls2/matematica-ce. 2012.

[56] Massimo Franceschet. «Who shall survive? Misure di centralità su reti sociali». http://www.uniud.it/didattica/facolta/scienze/la_facolta/piano-lauree-scientifiche-2010-2012/pls2/matematica-ce. 2012.

[79] Cleve Moler. *Experiments with MATLAB*. Freely available at http://www.mathworks.com/moler/index_exm.html. 2008.

[80] Cleve Moler. *Numerical computing with MATLAB*. Freely available at http://www.mathworks.com/moler/index_ncm.html. 2004.

[82] H Müntz. «Solution directe de l'équation séculaire et de quelques problémes analogues transcendants». In: *C. R. Acad. Sci. Paris* 156 (1913), pp. 43–46.

[83] H Müntz. «Sur la solution des équations séculaires et des équations intégrales». In: *C. R. Acad. Sci. Paris* 156 (1913), pp. 860–862.

[86] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[111] R von Mises e H Pollaczek-Geiringer. «Praktische Verfahren der Gleichungs-auflosung». In: *Z. Angew. Math. Mech.* 9 (1929), pp. 58–77.