Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Matematica

I MOTORI DI RICERCA: ALGORITMI A CONFRONTO E SPERIMENTAZIONE IN UNA CLASSE DI SCUOLA SUPERIORE

Tesi di Laurea in Calcolo Numerico e Didattica della Matematica

Relatore:
Chiar.mo Prof.
Giorgio Bolondi

Correlatore:
Chiar.ma Prof.
Carla Guerrini

Presentata da:
Sara Calandrini

I Sessione
Anno Accademico 2012/13

Introduzione

Questa è una tesi interdisciplinare che coniuga due importanti ambiti della Matematica: il Calcolo Numerico e la Didattica della Matematica.
L'argomento centrale è il *web information retrieval* ovvero quell'area dell'*information retrieval* composta dai motori di ricerca nel web. Il motore di ricerca di cui tratteremo maggiormente sarà Google. Il percorso che intraprenderemo può essere diviso in due parti: prima di tutto analizzeremo, da un punto di vista algebrico e computazionale, l'algoritmo originario che sta alla base del suddetto motore di ricerca, proponendone varie versioni e descrivendo brevemente le evoluzioni che ha avuto avuto; in secondo luogo, studieremo come affrontare tale tematica con una classe di scuola superiore e analizzeremo i risultati di due sperimenatazioni che ho svolto su tale argomento in due classi di quarta superiore. È proprio da questo duplice scopo (analisi dell'algoritmo e ideazione di un progetto per spiegare a dei ragazzi di scuola superiore il funzionamento di tale algoritmo) che deriva l'interdisciplinarietà della tesi. Per ottenere la maggior chiarezza possibile, ho suddiviso la tesi in due parti: la prima parte è strettamente relativa al Calcolo Numerico, mentre nella seconda verrà sviluppato ad analizzato il progetto di Didattica della Matematica. La prima parte è composta dai Capitoli 1, 2 e 3, mentre la seconda parte dai Capitoli 4, 5 e 6.
Il Capitolo 1 tratta dell'*information retrieval* e descrive due suoi metodi molto utilizzati: il *modello spazio-vettoriale* [2] ed una sua variante chiamata *Latent Semantic Indexing* (LSI) [4].
Nel Capitolo 2 inizieremo a parlare di *web information retrieval* e l'argomen-

to principale di tale capitolo sarà il Pagerank [4], [6]. Il concetto di Pagerank è alla base dell'algoritmo Google's Pagerank ideato da Brin e Page, inventori di Google. In questo capitolo analizzaremo tale algoritmo e ne proporremo varie versioni [7], [8].

Il Capitolo 3 tratta di altri due importanti metodi per il *web information retrieval*: HITS e SALSA [5], [6]. Entrambi i metodi presentano notevoli somiglianze e differenze con il metodo Pagerank.

Per quanto riguarda la seconda parte relativa alla Didattica della Matematica, il Capitolo 4 descrive la struttura del progetto svolto con le due classi di quarta superiore: vengono descritti gli obiettivi del progetto, la suddivisione dei vari argomenti lezione per lezione e le prove di valutazione somministrate.

Il Capitolo 5 riporta il 'mio diario di bordo': tratta di ciò che avvenuto nelle due classi in ogni singola lezione: vengono enunciati i concetti introdotti lezione per lezione e descritte le reazioni avute dai ragazzi insieme alle loro domande e dubbi.

Infine, nel Capitolo 6 vengono elaborate le conclusioni in merito a tale progetto. Si evidenzia quali obiettivi sono stati raggiunti e quali no dai ragazzi e viene effettuata una revisione del progetto, ovvero vengono evidenziati i suoi punti forti ed i suoi punti deboli e vengono proposte modifiche al fine di poterlo riproporre in altre classi.

Indice

Parte I

Capitolo 1

Con il termine *information retrieval* si indicano in genere metodi per estrarre informazioni da una collezione molto grande e spesso non strutturata di testi. Una applicazione tipica è la ricerca di un *abstract* o di una pubblicazione scientifica in un *database*. Per esempio in una applicazione medica si vogliono trovare tutti i lavori che trattano una particolare sindrome, per cui si formula una richiesta (*query*) con parole chiave che siano rilevanti per la sindrome in esame. Qui entra in gioco un sistema di rilevamento che fa corrispondere la richiesta ai documenti presenti nel *database* e riporta all'utente una lista di documenti che sono rilevanti, ordinati secondo la loro importanza. Prima dell'avvento dei moderni sistemi di calcolo, i ricercatori che avevano bisogno di una particolare informazione potevano cercare solo manualmente, per esempio in un catalogo a schede. Questi metodi manuali di indicizzazione presentano problemi, il primo dei quali è legato alla loro capacità. Ogni anno in Italia sono pubblicati circa 60000 libri mentre negli Stati Uniti circa 1.4 milioni e questi numeri sono piccoli se confrontati con il mondo digitale infatti, attualmente, ci sono circa 5 miliardi di pagine web su Internet. Un secondo problema riguarda la coerenza, infatti anche quando il numero di dati può essere gestito manualmente è difficile mantenere una coerenza negli indici creati dagli essere umani: l'estrazione di concetti e parole chiave può dipendere dalle esperienze ed opinioni di colui che organizza l'elenco. Questi problemi hanno alimentato lo sviluppo di tecniche automatiche di *information retrieval*. Quando implementati su sistemi

di computer ad alte prestazioni, questi metodi possono essere applicati a vasti *database* e possono, senza pregiudizio, creare modelli di associazione concetto-documento che costituiscono la struttura semantica della collezione di dati. Ma anche questi sistemi sono affetti da problemi: differenze fra il vocabolario degli autori del sistema e quello degli utenti pone delle difficoltà quando l'informazione viene processata senza l'intervento umano, la complessità di linguaggio e parole che possono avere diversi significati possono portare al ritrovamento di molti documenti irrilevanti. Proprio questi due ultimi aspetti (sinonimia e polisemia) sono due dei maggiori ostacoli per un metodo di indicizzazione. In questo capitolo vedremo come l'algebra lineare può essere usata nell'*information retrieval* e descriveremo due metodi molto utilizzati: il *modello spazio-vettoriale* [2] ed una sua variante chiamata Latent Semantic Indexing (LSI) [4].

1.1 Modello spazio-vettoriale

Nel *modello spazio-vettoriale* un vettore è usato per rappresentare ciascun articolo (o *documento*) in una collezione e ciascuna componente del vettore riflette una particolare parola chiave (o *termine*) associata al documento dato. Il valore assegnato ad ogni componente rispecchia l'importanza del termine nel rappresentare il contenuto del documento e tipicamente tale valore è una funzione della frequenza con cui il termine appare nel documento o nella collezione totale dei documenti. Un *database* che contiene d documenti descritti da t termini è rappresentato da una matrice A di ordine $t \times d$ chiamata *matrice termine-documento*. I d vettori che rappresentano i d documenti formano le colonne della matrice, per cui sono chiamati *vettori-documento*, mentre le righe di A sono chiamate *vettori-termine*. Quello che è importante, dal punto di vista dell'*information retrieval*, è che possiamo sfruttare le relazioni geometriche fra i *vettori-documento* per modellare somiglianze e differenze nel contenuto dei documenti ed inoltre possiamo confrontare geometricamente i *vettori-termine* per identificare somiglianze e differenze nell'uso dei termini.

Per quanto riguarda gli elementi della matrice A, essi sono spesso rappresentati come prodotto di due valori: $a_{i,j} = l_{i,j}\, g_i$. Il fattore g_i è un fattore di peso globale cioè riflette il valore complessivo del termine i come termine di indicizzazione per l'intera collezione. Per esempio, consideriamo un termine molto comune come computer all'interno di una collezione di articoli su personal computers. Non è importante includere questo termine nella descrizione di un documento dato che sappiamo che tutti i documenti trattano di personal computers, cosí è opportuno assegnare a g_i un valore piccolo. Il fattore $l_{i,j}$ è un fattore di peso locale cioè riflette l'importanza del termine i all'interno del documento j. Siccome un documento generalmente usa solo un piccolo sottoinsieme dell'intero dizionario di termini generato per un certo *database*, la maggior parte degli elementi di una *matrice termine-documento* sono zero, quindi la matrice è sparsa.

Quando un utente interroga il *database* per trovare documenti rilevanti, in qualche modo usa la rappresentazione di questi documenti come elementi di uno spazio vettoriale. La richiesta fatta dall'utente è un insieme di termini ed è rappresentabile proprio come un documento. È probabile che molti termini del *database* non appaiano nella richiesta per cui molte componenti del vettore *query* saranno zero. Per determinare i documenti che sono rilevanti per quella particolare richiesta si usa il *query matching* che consiste nel trovare i documenti che sono più simili alla richiesta nell'uso e nel peso dei termini. Nel *modello spazio-vettoriale* i documenti selezionati sono quelli che geometricamente sono più vicini alla *query* secondo una qualche misura ed una misura di somiglianza molto comune è il coseno dell'angolo fra la *query* ed i *vettori-documento*. Se indichiamo con a_j le colonne della *matrice termine-documento* A, dove $j = 1, \ldots, d$, e con q la *query* i d coseni possono essere calcolati tramite la seguente formula:

$$cos\,\theta_j = \frac{{a_j}^T q}{||a_j||_2\, ||q||_2} = \frac{\sum_{i=1}^{t} a_{i,j} q_i}{\sqrt{\sum_{i=1}^{t} a_{i,j}^2}\, \sqrt{\sum_{i=1}^{t} q_i^2}} \tag{1.1}$$

per $j = 1, \ldots, d$. Dato che la *query* ed i *vettori-documento* sono tipicamente sparsi, il prodotto scalare e le norme sono generalmente poco costose da cal-

colare. Inoltre le norme dei *vettori-documento* $||a_j||_2$ devono essere calcolate solo una volta per ogni data *matrice termine-documento*.
Diciamo che un documento a_j è giudicato rilevante se

$$cos\,\theta_j = \frac{{a_j}^T q}{||a_j||_2\,||q||_2} > tol \quad ,$$

dove *tol* è una tolleranza fissata. Se si sceglie una tolleranza bassa più documenti vengono ritrovati e ciò può essere una cosa positiva, ma si corre il rischio che troppi documenti, che possono non essere rilevanti, vengano riportati. Una misura delle performance del modello è data da:

$$Precisione: \quad P = \frac{D_r}{D_t} \quad ,$$

dove D_r è il numero di documenti rilevanti trovati e D_t è il numero totale di documenti trovati, e da

$$Recall: \quad R = \frac{D_r}{N_r} \quad ,$$

dove N_r è il numero totale di documenti rilevanti nel *database*. Se viene scelto un valore grande per la tolleranza *tol* avremo una *Precisione* grande ma un *Recall* basso, mentre per un valore piccolo di *tol* avremo un *Recall* alto ed una bassa *Precisione*.

Esempio 1.1.1. Consideriamo una semplice collezione di 5 titoli descritti da 6 termini. Dato che il contenuto di un documento è determinato dalla frequenza relativa dei termini e non dal numero totale di volte che quel particolare termine appare, gli elementi della matrice verranno scalati in modo che la norma 2 di di ciascuna colonna sia 1. La scelta dei termini usati per descrivere il *database* determina non solo la sua dimensione ma anche la sua utilitá. In questo esempio useremo solo termini strettamente correlati alla cucina. Ricordiamo che, prima di costruire la *matrice termine-documento*, devono essere fatti due preprocessamenti sui termini: eliminazione delle *stop words* e *stemming*. Le *stop words* sono quelle parole che si trovano in tutti i documenti e che non provocano la distinzione di un documento da un altro: *about, above, accordingly, across, after, afterwards, again, against, all,*

allows, alone, along, already, also, although, always, among, an, and, ... *Stemming* è il processo di ridurre le parole che sono coniugate ad un suffisso, nell'esempio sottostante nessuna informazione viene persa nella riduzione:

$$\left.\begin{array}{r} computable \\ computation \\ computing \\ computed \\ computational \end{array}\right\} \rightarrow \; comput\ .$$

I 5 documenti (titoli) che consideriamo sono:
D1: How to Bake Bread Bread Without Recipes.
D2: The Classic Art of Viennese Pastry.
D3: Numerical Recipes: the Atr of Scientific Computing.
D4: Breads, Pastries, Pies and Cakes: Quantity Baking Recipes.
D5: Pastry: a Book of Best French Recipes.

I 6 termini che consideriamo sono:
T1: bak(e, ing),
T2: recipes,
T3: bread,
T4: cake,
T5: pastr(y, ies),
T6: pie.

La *matrice termine-documento* di ordine 6×5 ha la seguente forma:

$$\hat{A} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \ .$$

Scalando gli elementi, in modo che la norma 2 di di ciascuna colonna sia 1, otteniamo:

$$A = \begin{pmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{pmatrix} .$$

Supponiamo che un utente che cerca informazioni in ambito culinario inizi una ricerca per libri su *baking bread*. Il corrispondente vettore *query* assume la seguente forma:

$$q^{(1)} = (1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)^T$$

ed ha componenti diverse da zero per i termini *baking* e *bread*. Ora, per il *query matching*, non ci resta che calcolare i coseni degli angoli θ_j fra il vettore *query* $q^{(1)}$ e i *vettori-documento* a_j utilizzando la formula (1.1). Siccome il nostro esempio ha dimensioni piccole come valore della tolleranza *tol* possiamo prendere 0.5, quindi un documento sarà considerato rilevante se il coseno dell'angolo che crea con il vettore *query* è maggiore di 0.5. Per la richiesta $q^{(1)}$ gli unici coseni diversi da zero sono $cos\,\theta_1 = 0.8165$ e $cos\,\theta_4 = 0.5774$ e tutti i documenti che trattano di *baking bread* (il primo ed il quarto) sono considerati rilevanti. Il secondo, il terzo ed il quinto documento, che non trattano nessuno dei due *topics*, sono correttamente ignorati.
Se l'utente semplifica la sue richiesta e vuole ottenere informazioni solo su *baking* i risultati saranno molto diversi. In questo caso il vettore *query* è

$$q^{(1)} = (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T \ ,$$

e i coseni degli angoli fra la *query* e i cinque vettori documenti sono, in ordine, 0.5774, 0, 0, 0.4082, 0. Solo $cos\,\theta_1$ supera la tolleranza prefissata ed il quarto documento, che è il più attinente alla *query*, non è considerato rilevante. Gli studiosi hanno cercato di superare questo problema proponendo nuove tecniche di rappresentazione dei dati con la *matrice termine-documento*.

Noi ci soffermeremo su una tecnica particolare che sostituisce l'esatta *matrice termine-documento* con una sua approssimazione di rango più basso. Questa approssimazione può essere ottenuta in diversi modi e noi ne vedremo due: il primo sfrutta la fattorizzazione QR [2], il secondo la decomposizione in valori singolari (SVD) utilizzata nel metodo LSI [4].

1.2 Fattorizzazione QR

Vediamo come la fattorizzazione QR può essere usata per identificare e rimuovere informazioni nella rappresentazione matriciale del *database*. Dal punto di vista dell'algebra lineare, quello che facciamo è trovare il rango della *matrice termine-documento* e poi ridurlo. Il processo che porta alla riduzione del rango può essere suddiviso in vari passi. Il primo passo consiste nell'identificare la dipendenza fra le colonne e le righe della *matrice termine-documento*. Se consideriamo lo spazio generato dalle colonne di una matrice A $t \times d$ di rango r_A possiamo individuare un insieme di vettori base calcolando la fattorizzazione QR:

$$A = QR\,,$$

dove R è una matrice triangolare superiore di ordine $t \times d$ e Q è una matrice ortogonale di ordine $t \times t$, cioè $Q^T Q = Q\,Q^T = I$. Questa fattorizzazione esiste per una qualsiasi matrice A. La relazione $A = QR$ mostra che tutte le colonne di A sono combinazioni lineari delle colonne di Q, cosí un sottoinsieme di r_A colonne di Q forma una base per lo spazio delle colonne di A.
Ora ritorniamo all'esempio (1.1.1) ed applichiamo alla matrice A la fattorizzazione QR. La *matrice termine-documento* ha rango 4 dato che la colonna 5 è la somma delle colonne 2 e 3. Le matrici che otteniamo, attraverso la

fattorizzazione, hanno la seguente forma:

$$Q = \begin{pmatrix} -0.5774 & 0 & -0.4082 & 0 & -0.7071 & 0 \\ -0.5774 & 0 & 0.8165 & 0 & 0 & 0 \\ -0.5774 & 0 & -0.4082 & 0 & 0.7071 & 0 \\ 0 & 0 & 0 & -0.7071 & 0 & -0.7071 \\ 0 & -1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.7071 & 0 & 0.7071 \end{pmatrix} , \quad (1.2)$$

$$R = \begin{pmatrix} -1.0001 & 0 & -0.5774 & -0.7071 & -0.4082 \\ 0 & -1.0000 & 0 & -0.4082 & -0.7071 \\ 0 & 0 & 0.8165 & 0 & 0.5774 \\ 0 & 0 & 0 & -0.5774 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} . \quad (1.3)$$

A questo punto riscriviamoci la fattorizzazione $A = QR$ come

$$A = (Q_A \quad Q_A^{\perp}) \begin{pmatrix} R_A \\ 0 \end{pmatrix} = Q_A R_A + Q_A^{\perp} \cdot 0 = Q_A R_A \ , \quad (1.4)$$

dove Q_A è la matrice di ordine 6×4 formata dalle prime quattro colonne di Q, $Q_A^{\perp}$ è la rimanente sottomatrice di ordine 6×2 e R_A è costituita dalle righe di R diverse da zero (le prime quattro). Questa partizione rivela che le colonne di $Q_A^{\perp}$ non contribuiscono alla costruzione di A e che il rango di A, R e R_A sono uguali, cosí le quattro colonne di Q_A costituiscono la base per lo spazio generato dalle colonne di A. È importante notare che la partizione di R in due sottomatrici, di cui una costituita tutta da zeri, è una caratteristica di questa particolare matrice A. In generale, è necessario usare il pivoting per colonne durante la fattorizzazione QR per assicurare che gli zeri appaiano nella parte inferiore della matrice. Quando il pivoting è usato la fattorizzazione diventa $AP = QR$, dove P è una matrice di permutazione. Il pivoting serve per ordinare in modo che le prime r_A colonne di Q formino una base per lo spazio delle colonne della matrice A e le corrispondenti righe della matrice

R forniscano i coefficienti per le combinazioni lineari di questi vettori base, combinazioni che costituiscono le colonne di A. Notiamo che le colonne di $Q_A^{\perp}$ costituiscono una base per il complemento ortogonale dello spazio delle colonne di AP e, di conseguenza, dello spazio delle colonne di A. Il pivoting per colonne fornisce importanti vantaggi numerici senza modificare il *database* in quanto permutare le colonne di A porta solo ad un riordinamento dei *vettori-documento*. D'ora in poi, indicheremo la matrice AP semplicemente con A per maggiore chiarezza.
Per quanto riguarda il *query matching*, esso procede con i fattori Q ed R al posto della matrice A. I coseni degli angoli θ_j fra il vettore *query* q e i *vettori-documento* a_j sono dati da

$$cos\,\theta_j = \frac{a_j^T\, q}{||a_j||_2\, ||q||_2} = \frac{(Q_A\, r_j)^T\, q}{||Q_A\, r_j||_2\, ||q||_2} = \frac{r_j^T\, (Q_A^T\, q)}{||r_j||_2\, ||q||_2} \tag{1.5}$$

per $j = 1, \ldots, d$. In questa relazione abbiamo usato il fatto che $||Q_A\, r_j||_2 = ||r_j||_2$ in quanto Q e quindi Q_A sono matrici ortogonali.
Ritornando all'esempio (1.1.1) ed utilizzando nuovamente il vettore *query* $q^{(1)}$ (*baking bread*) osserviamo che non c'è perdita di informazioni utilizzando la forma fattorizzata. Infatti i coseni calcolati attraverso la (1.5) sono identici a quelli calcolati con la (1.1): 0.8165, 0, 0, 0.5774 e 0.
Prima di passare alla costruzione di un'approssimazione low-rank della *matrice termine-documento*, osserviamo che la rappresentazione partizionata vista in (1.4) ci permette di formulare un'interpretazione geometrica della procedura del *query matching*. Notiamo che, per la matrice ortogonale Q

$$I = QQ^T = (Q_A \quad Q_A^{\perp})(Q_A \quad Q_A^{\perp})^T = Q_A Q_A^T + Q_A^{\perp}(Q_A^{\perp})^T \quad ,$$

quindi possiamo scrivere il vettore *query* q come la somma delle sue componenti nello spazio delle colonne di A e nel complemento ortogonale di tale spazio come segue:

$$\begin{aligned} q = Iq = QQ^T &= [Q_A Q_A^T + Q_A^{\perp}(Q_A^{\perp})^T]\, q = \\ &= Q_A Q_A^T\, q + Q_A^{\perp}(Q_A^{\perp})^T q = q_A + q_A^{\perp} \quad . \end{aligned} \tag{1.6}$$

La componente $q_A = Q_A^{\perp}(Q_A^{\perp})^T q$ è chiamata *proiezione ortogonale* di q nello spazio delle colonne di Q_A in quanto è la più vicina approssimazione del vettore *query* q nello spazio delle colonne di A. Più precisamente

$$||q-q_A||_2 = \min \{||q-x||_2, \text{ con } x \text{ appartenente allo spazio delle colonne di } A\} .$$

Dimostrazione. Se i vettori q_A e x sono entrambi nello spazio delle colonne di A anche il vettore $q_A - x$ ci sta. Il vettore $q - q_A \equiv q_A^{\perp}$ è ortogonale a qualsiasi vettore in questo spazio (per definizione) ed usando il teorema di Pitagora emerge che:

$$||q - x||_2^2 = ||q - q_A + q_A - x||_2^2 = ||q - q_A||_2^2 + ||q_A - x||_2^2 \geq ||q - q_A||_2^2 \ .$$

A questo punto, sostituendo (1.6) in (1.5), vediamo che solo la componente q_A contribuisce veramente nel prodotto scalare usato per calcolare i coseni fra la *query* ed i *vettori-documento*:

$$cos\,\theta_j = \frac{a_j^T q_A + a_j^T\, q_A^{\perp}}{||a_j||_2\,||q||_2} = \frac{a_j^T q_A + a_j^T\, Q_A^{\perp}(Q_A^{\perp})^T\, q}{||a_j||_2\,||q||_2} \ .$$

Dato che a_j è una colonna di A, essa è ortogonale alle colonne di $Q_A^{\perp}$ e ciò implica che $a_j^T Q_A^{\perp} = 0$, per cui nella formula dei coseni abbiamo:

$$cos\,\theta_j = \frac{a_j^T q_A + 0 \cdot (Q_A^{\perp})^T\, q}{||a_j||_2\,||q||_2} = \frac{a_j^T q_A}{||a_j||_2\,||q||_2} \ .$$

Un'interpretazione di questo risultato è che la richiesta imperfetta dell'utente è automaticamente sostituita nel prodotto scalare con la sua migliore approssimazione proveniente dal contento del *database*. La componente $q_A^{\perp}$, che non condivide il contenuto con nessuna parte dello spazio delle colonne di A, è ignorata. Sfruttando questa osservazione, possiamo sostituire q con la sua proiezione ed ottenere una nuova misura di somiglianza:

$$cos\,\theta_j{}' = \frac{a_j^T q_A}{||a_j||_2\,||q||_2} \ , \tag{1.7}$$

cioè confrontiamo la proiezione della richiesta dell'utente con i *vettori-documento*. Per un dato indice j i due coseni sono in relazione fra di loro:

$$cos\,\theta_j = cos\,\theta_j{}' \frac{||q_A||_2}{||q||_2} = cos\,\theta_j{}' \frac{||q_A||_2}{\sqrt{||q_A||_2^2 + ||q_A^{\perp}||_2^2}} \ . \tag{1.8}$$

Poichè il fattore $\frac{||q_A||_2}{\sqrt{||q_A||_2^2 + ||q_A^\perp||_2^2}}$ è limitato superiormente da 1, i coseni calcolati usando q sono sempre minori o uguali ai coseni calcolati usando q_A. Come risultato abbiamo che un vettore *query* quasi ortogonale allo spazio delle colonne di A è più probabile che sia giudicato rilevante quando usiamo q_A piuttosto che quando usiamo q, anche se tale vettore ha solo una piccola componente in questo spazio. In altre parole, usare la formula (1.7) potrebbe aiutare ad identificare più documenti rilevanti ma potrebbe anche portare all'aumento del numero di documenti irrilevanti.

1.3 Approssimazione low-rank

La fattorizzazione QR ci fornisce un mezzo per affrontare le incertezze del *database*, infatti il processo di indicizzazione del *database* può portare ad incertezze nella *matrice termine-documento*. Un *database* e la sua rappresentazione matriciale possono essere costruiti su un lungo periodo di tempo, da tante persone con diverse esperienze e differenti opinioni su come suddividere per categorie il contenuto del *database*. Di conseguenza, una *matrice termine-documento* potrebbe essere meglio rappresentata da una somma di matrici $A + E$ dove E è la matrice che rappresenta le incertezze ed i suoi valori riflettono le informazioni mancanti o incomplete sui documenti o anche le differenti opinioni sull'importanza di documenti che trattano un determinato argomento. Ora, se accettiamo il fatto che la nostra matrice A è solo un rappresentante di una grande famiglia di matrici relativamente simili che rappresentano il *database*, è ragionevole chiedersi se ha senso cercare di determinare il suo rango esatto. Per esempio, se scopriamo che la nostra matrice ha rango r_A e, usando l'algebra lineare, concludiamo che modificare A con l'aggiunta di una piccola incertezza E porterebbe ad una matrice $A + E$ di rango k con $k < r_A$, poi potremmo dimostrare che il nostro problema si può rappresentare con una matrice di rango k e che lo spazio delle colonne di A non è necessariamente la miglior rappresentazione del contenuto semantico del *database*. Vediamo come ridurre il rango può aiutare a rimuovere

informazioni estranee o rumore (*noise*) dalla rappresentazione matriciale del *database*. Prima di tutto abbiamo bisogno di una nozione di *dimensione* di una matrice ed in particolare abbiamo bisogno di sapere quando una matrice è *piccola* in confronto ad un'altra. Se generalizziamo il concetto di norma euclidea di un vettore alle matrici, il risultato è la cosiddetta norma di Frobenius che, per una matrice X di ordine $t \times d$, è definita come:

$$||X||_F = \sqrt{\sum_{i=1}^{t}\sum_{j=1}^{d} x_{i,j}^2} \ . \tag{1.9}$$

La norma di Frobenius può essere definita anche tramite la traccia della matrice X^TX ovvero: $||X||_F = \sqrt{\text{Traccia}(X^TX)} = \sqrt{\text{Traccia}(XX^T)}$.
Usando la seconda definizione si ha che moltiplicando a sinistra la matrice X con una matrice O di ordine $t \times t$ la norma di Frobenuis non cambia:

$$\begin{aligned} ||OX||_F &= \sqrt{\text{Traccia}((OX)^T(OX))} = \sqrt{\text{Traccia}(X^TO^TOX)} = \\ &= \sqrt{\text{Traccia}(X^TX)} = ||X||_F \ . \end{aligned}$$

Ricordiamoci che il nostro scopo è trovare una approssimazione di basso rango della matrice A. Focalizziamoci sulla matrice triangolare superiore R, ricordando che il rango di A è uguale al rango di R. Il rango di R è facile determinare, infatti è uguale al numero di elementi della diagonale principale che sono diversi da zero. La fattorizzazione QR con pivoting ci aiuta a manipolare il rango di R dato che tende a separare gli elementi lontani da zero da quelli vicini a zero, ovvero spinge le componenti più grandi (in modulo) verso l'angolo in alto a sinistra della matrice e le componenti più vicine a zero verso l'angolo in basso a destra. Per esempio la matrice R dell'esempio (1.1.1) può essere partizionata come segue:

$$R = \left(\begin{array}{ccc|cc} -1.0001 & 0 & -0.5774 & -0.7071 & -0.4082 \\ 0 & -1.0000 & 0 & -0.4082 & -0.7071 \\ 0 & 0 & 0.8165 & 0 & 0.5774 \\ \hline 0 & 0 & 0 & -0.5774 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \ .$$

Con questa suddivisione la sottomatrice R_{22} risulta una piccola parte della matrice R; nello specifico $\frac{||R_{22}||_F}{||R||_F} = \frac{0.5774}{2.2361} = 0.2582$. Ora creiamo una nuova matrice triangolare superiore $\hat{R}$ ponendo la matrice R_{22} uguale ad una matrice di tutti zeri. La nuova matrice $\hat{R}$ ha rango 3 e quindi anche la matrice $A+E = Q\hat{R}$ avrá lo stesso rango. La matrice E che rappresenta le incertezze è data dalla differenza

$$E = (A+E) - A = Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} - Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = Q\begin{pmatrix} 0 & 0 \\ 0 & -R_{22} \end{pmatrix} .$$

Notiamo che: $||E||_F = \left|\left|\begin{pmatrix} 0 & 0 \\ 0 & -R_{22} \end{pmatrix}\right|\right|_F = ||R_{22}||_F$.

Poichè $||A||_F = ||R||_F$, abbiamo che $\frac{||E||_F}{||A||_F} = \frac{||R_{22}||_F}{||R||_F} = 0.2582$. In altre parole, fare un cambiamento relativo del 26% nella matrice R porta ad una cambiamento delle stesse dimensioni nella matrice A e questa variazione riduce di 1 il rango di entrambe le matrici. Cosí potremmo ritenere accettabile per il *query matching* usare l'approssimazione di rango 3 $A+E$ al posto dell'originale *matrice termine-documento*. Se calcoliamo i coseni usando la formula (1.5) non abbiamo bisogno di calcolare la matrice $A+E$ esplicitamente ma piuttosto possiamo usare, dalla sua fattorizzazione QR, le prime tre colonne di Q e la matrice triangolare R che ha tre righe di tutti zero. Per verificare che non abbiamo causato una perdita di accuratezza , ritorniamo all'esempio (1.1.1) usando al matrice $A+E$ al posto dell'originale *matrice termine-documento* A. I coseni calcolati per la richiesta $q^{(1)}$ (*baking bread*) sono: 0.8165, 0, 0, 0.7071 e 0, mentre i coseni calcolati per la *query* $q^{(2)}$ sono 0.5774, 0, 0, 0.5000 e 0. In entrambi i casi i risultati sono veramente migliorati quindi la nostra approssimazione di rango 3 $A+E$ sembra essere una rappresentazione migliore del nostro *database* rispetto all'originale *matrice termine-documento*. Per ottenere una riduzione ancora maggiore, partizioniamo la matrice R in modo che anche la sua terza riga e la sua terza colonna siamo incluse in R_{22}. In questo caso, $\frac{||R_{22}||_F}{||R||_R} = 0.5146$ e scartando R_{22} per creare una approssimazione di rango 2 della *matrice termine-documento* in-

troduciamo un cambiamento relativo del 52% in questa matrice. I coseni calcolati per $q^{(1)}$ sono ora: 0.8165, 0, 0.8165, 0.7071 e 0.4082, mentre per $q^{(2)}$ sono 0.5774, 0, 0.5774, 0.5000 e 0.2887. In entrambi i casi, qualche documento irrilevante è erroneamente identificato quindi un cambiamento del 52% in R e A è inaccettabile perché troppo grande. In generale, non è possibile spiegare perché una variante della *matrice termine-documento* è migliore di un'altra per una data *query*. Comunque, abbiamo visto che è possibile migliorare la performance del metodo riducendo il rango della *matrice termine-documento*. Notiamo che anche il cambiamento del 26% che abbiamo considerato accettabile nel nostro esempio è piuttosto grande nel contesto delle applicazioni scientifiche o ingegneristiche dove è solitamente richiesta un'accuratezza di tre o più cifre decimali.

1.4 LSI

Il metodo LSI (*Latent Semantic Indexing*) è una variante del *modello spazio-vettoriale* e si basa sull'assunzione che ci sono delle strutture semantiche latenti che sono sottese ai dati e che sono perturbate dalla grande varietà di parole usate. Questa struttura latente può essere scoperta ed evidenziata proiettando i dati (cioè la *matrice termine-documento* e la *query*) su uno spazio di dimensioni ridotte usando la SVD. Più precisamente si tenta di superare i problemi del *matching* lessicale utilizzando indici concettuali di derivazione statistica. Il metodo LSI assume che esista una struttura base (o latente) nell'uso delle parole che risulta parzialmente oscurata dall'uso e dalla variabilità dei termini che vengono scelti. Per stimare la struttura nelle parole usate nei documenti si può utilizzare la scomposizione in valori singolari troncata (TSVD). Il *retrieval* sarà eseguito usando il *database* dei valori e vettori singolari ottenuti dalla TSVD. La mole di tempo necessaria per il processamento del metodo LSI è dato dal tempo speso nel calcolo della TSVD di matrici di grandi dimensioni e sparse. Per l'implementazione del metodo occorre, ancora una volta, costruire una *matrice termine-documento*, i cui ele-

menti sono le occorrenze di ciascuna parola in un particolare documento, cioè $A = [a_{ij}]$ dove a_{ij} indica la frequenza con cui il termine chiave i compare nel documento j. Poichè ogni termine esistente non compare in ogni documento la matrice A è generalmente sparsa. Come nel *modello spazio-vettoriale*, si usano pesi di tipo globale e locale per aumentare o diminuire l'importanza di un termine entro o tra i documenti: $a_{ij} = L_{ij}G_i$, dove L_{ij} è il peso locale per il termine i nel documento j e G_i è il peso globale per il termine i. La matrice A viene fattorizzata in tre matrici mediante la decomposizione in valori singolari:

$$A = U\Sigma V^T \ ,$$

dove U e V sono due matrici ortogonali nelle cui colonne sono memorizzati gli autovettori singolari sinistri e destri rispettivamente e Σ è una matrice diagonale avente i valori singolari σ_i di A, in ordine decrescente, sulla sua diagonale. Questa fattorizzazione esiste per una qualunque matrice A.
Ricordiamo che il rango r_A della matrice A è uguale al numero di valori singolari diversi da zero e che la $||A||_F$ può essere definita in termini di questi valori singolari:

$$||A||_F = ||U\Sigma V^T||_F = ||\Sigma V^T||_F = ||\Sigma||_F = \sqrt{\sum_{j=1}^{r_A} \sigma_j^2} \ .$$

Ci sono diverse somiglianze fra la SVD ($A = U\Sigma V^T$) e la fattorizzazione QR ($AP = QR$). In primo luogo, il rango r_A della matrice A è uguale sia al numero di elementi diagonali di R diversi da zero sia al numero di elementi diagonali di Σ diversi da zero. Inoltre, sia le prime r_A colonne di Q che le prime r_A colonne di U sono una base per lo spazio della colonne di A, ma una grande differenza fra le due fattorizzazioni risiede nelle proprietà delle loro approssimazioni A_k. Le matrici U, Σ e V, che si ottengono tramite la SVD, riflettono la suddivisione delle relazioni originali in vettori linearmente indipendenti che vengono chiamati in questo caso *factor values*. I k fattori o meglio le k-triplette singolari s_i, u_i, v_i (con $i = 1, \ldots, k$) più grandi permettono di approssimare la *matrice* originale *termine-documento* con una

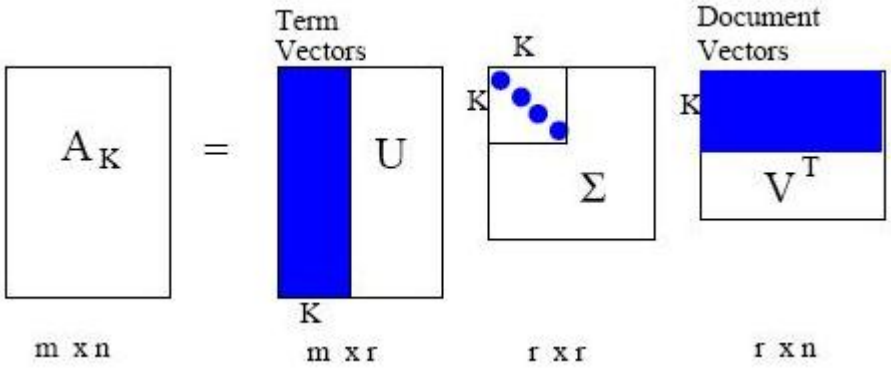
matrice A_k di rango k che ha la seguente forma

$$A_k = \sum_{j=1}^{k} \sigma_i u_i v_i^T \quad .$$

Un importante teorema (di Eckart e Joung) dimostra che tale matrice A_k minimizza la distanza fra A e le sue approssimazioni di rango k ovvero:

$$||A - A_k||_F = \min_{rango(X) \leq k} ||A - X||_F = \sqrt{\sigma_{k+1}^2 + \ldots + \sigma_{r_A}^2} \quad .$$

In un certo senso l'SVD può essere vista come una tecnica per ricavare un insieme di indici di variabili o fattori non correlati attraverso i quali ciascun termine e ciascun documento è rappresentato da un vettore nel k-spazio usando elementi dei vettori singolari destri e sinistri.



Risulta importante, per il metodo LSI, che la matrice A_k non ricostruisca la *matrice termine-documento* in modo esatto. In questo modo la TSVD cattura la maggior parte della struttura implicita, ma rilevante per l'associazione termini-documenti, e allo stesso tempo rimuove il rumore o la variabilità dovuta all'uso delle parole che affliggono il *retrieval* basato sul *matching* esatto dei caratteri delle parole. Intuitivamente, siccome il numero delle dimensioni k è molto più piccolo del numero dei termini, saranno ignorate meno le differenze nella terminologia. In questo modo termini che si presentano in documenti simili saranno più vicini l'un l'altro nello spazio k-dimensionale anche se non si presentano nello stesso documento. Ciò significa che alcuni

documenti che non condividono alcune parole con la *query* possono ciononostante essere vicini nel k-spazio. Per esempio consideriamo le parole *car, automobile, driver* ed *elephant*. *Car* e *automobile* sono sinonimi, *driver* è in relazione con *car* e *automobile* mentre *elephant* non ha collegamenti con nessuno. Nella maggior parte dei sistemi di *retrieval* la ricerca *automobile* non è detto che riporti, con più probabilità, documenti che riguardano *cars* rispetto a documenti che contengono *elephant*, se la parola *automobile* non viene usata nel documento. Per questi motivi è preferibile che la *query* riguardo *automobile* ritrovi anche articoli sulle *cars* e anche articoli su *drivers* con una portata inferiore. Le parole *cars* e *automobile* appariranno in documenti in cui vi sono molte medesime parole quali *motors, model, vehicle ,carmakers, ecc ...*, il contesto per *driver* si sovrapporrà con una portata inferiore ed infine i documenti relativi ad *elephant* saranno decisamente dissimili.
In conclusione, l'idea di base del metodo LSI è di modellare in modo esplicito la interrelazione fra i termini e di sfruttare ciò per migliorare il *retrieval*.
Approssimiamo A con una matrice di rango k:

$$A \approx U_k \Sigma_k V_k^T = U_k H_k \ .$$

Le colonne di U_k stanno nello spazio delle colonne di A e ne formano una base ortonormale, quindi le utilizziamo per approssimare i documenti. Consideriamo H_k costituita dai suoi vettori colonna $H_k = (h_1, h_2, \ldots, h_n)$. Dal fatto che $A \approx U_k H_k$ abbiamo che $a_j = U_k h_j$, che significa che la colonna j di H_k ha in sè le coordinate del documento j in termini della base ortogonale. Allora in questa approssimazione di rango k la *matrice termine-documento* è rappresentata da $A_k = U_k H_k$ e per il *query matching* calcoliamo:

$$q_k = q^T A_k = q^T U_k H_k = (U_k^T q)^T H_k \ ,$$

cioè calcoliamo le coordinate della *query* in termini della nuova base e tali coordinate le indichiamo con q_k. Il coseno quindi sarà dato da:

$$cos\,\theta_j'' = \frac{q_k^T h_j}{||q_k||_2 \, ||h_j||_2} \tag{1.10}$$

e questo significa che il *query matching* è eseguito nello spazio k-dimensionale. In questo modo l'LSI risulta più efficiente nel recupero delle informazioni. Inoltre occorre sottolineare che, comunemente, la maggior parte delle matrici termine-documento sono ben condizionate cioè non ci sono dei salti nella sequenza dei loro valori singolari (è stato dimostrato sperimentalmente). Se si calcola l'errore di approssimazione di A_k con k piccolo questo risulta molto alto in contrapposizione al fatto che si migliorano le prestazioni del retrieval. È interessante vedere quali sono le direzioni più importanti nei dati. Dalla teoria sulla SVD sappiamo che i primi vettori singolari sinistri sono le direzioni dominanti nello spazio delle colonne di A le loro componenti più grandi dovrebbero indicare quali sono queste direzioni. Uno studio sistematico dei differenti aspetti dell'LSI ha dimostrato che esso migliora le prestazioni per piccoli valori del rango k, in modo sorprendente, e allo stesso tempo si hanno errori molto alti nell'approssimazione della matrice A. Non è possibile provare in modo sistematico che l'LSI migliori l'efficienza del *retrieval*, ma comunque ciò risulta in molti esperimenti.
Ora applichiamo l'LSI al nostro esempio (1.1.1) e vediamo quali documenti saranno considerati rilevanti. Come tolleranza per il coseno prendiamo nuovamente il valore 0.5. Applicando la SVD alla matrice A otteniamo:

$$U = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 & -0.2847 & -0.7071 & 0 \\ 0.7479 & -0.3981 & -0.5249 & 0.0816 & 0 & 0 \\ 0.2670 & -0.2567 & 0.5308 & -0.2847 & 0.7071 & 0 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 & 0 & -0.7071 \\ 0.5198 & 0.8423 & 0.0838 & -0.1158 & 0 & 0 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 & 0 & 0.7071 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1.6950 & 0 & 0 & 0 & 0 \\ 0 & 1.1158 & 0 & 0 & 0 \\ 0 & 0 & 0.8403 & 0 & 0 \\ 0 & 0 & 0 & 0.4195 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$V = \begin{pmatrix} 0.4366 & -0.4717 & 0.3688 & -0.6715 & 0 \\ 0.3067 & 0.7549 & 0.0998 & -0.2760 & -0.5000 \\ 0.4412 & -0.3568 & -0.6247 & 0.1945 & -0.5000 \\ 0.4909 & -0.0346 & 0.5711 & 0.6571 & 0 \\ 0.5288 & 0.2815 & -0.3712 & -0.0577 & 0.7071 \end{pmatrix}.$$

A è una matrice di rango 4 dato che ha 4 valori singolari diversi da 0; le ultime due righe di Σ costituite da tutti zeri ci dicono che le prime 4 colonne di U formano una base per lo spazio delle colonne di A.
Ponendo $k = 3$ commettiamo un errore relativo $\left(\frac{||A - A_3||_F}{||A||_F}\right)$ pari a 0.18692 quindi effettuiamo un cambiamento del 19%. I coseni calcolati per la richiesta $q^{(1)}$ (*baking bread*) e per la richiesta $q^{(2)}$ (*baking*), tramite la formula (1.10), sono i medesimi ovvero: 0.840, -0.239, 0.223, 0.733 e -0.009. Solo il primo ed il quarto valore superano la tolleranza e ciò è proprio quello che volevamo infatti per entrambe le *queries* i documenti 1 e 4 risultano rilevanti.
Se scegliamo $k = 2$ l'errore relativo cresce a 0.383 quindi si ha un cambiamento del 38%. I coseni calcolati sono: 0.968, -0.302, 0.713, 0.713 e 0.112 di nuovo per entrambe le *queries*. Purtroppo in questo caso anche il coseno relativo al documento 3 supera la tolleranza fissata quindi tale documento viene considerato rilevante anche se in realtá non lo è. Notiamo che le variazioni del 19% e 38% richieste per ridurre il rango della matrice tramite l'LSI sono minori rispetto alle corrispondenti variazioni richieste dalla fattorizzazione QR ma il risultato a cui arriviamo è lo stesso: la miglior approssimazione per il nostro *database* si ha con un rango pari a 3.

Capitolo 2

Pagerank

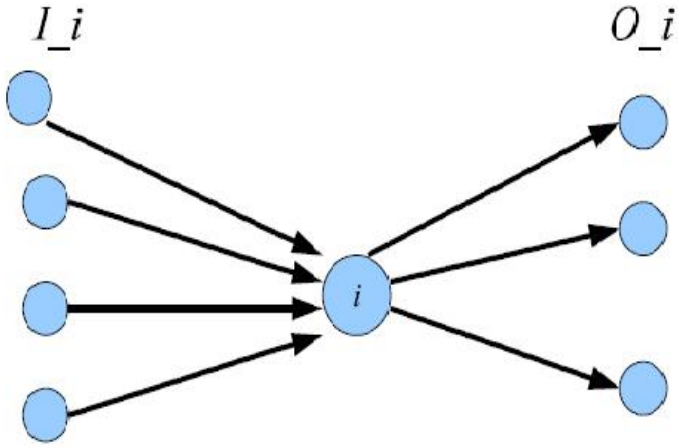
Un'area ben conosciuta dell'*information retrieval* è composta dai motori di ricerca nel web dove generalmente la frase che viene cercata è breve e spesso ci sono talmente tanti documenti rilevanti che non è possibile presentarli tutti insieme all'utente. In questa applicazione, l'ordine del risultato della ricerca è cruciale per l'efficienza del motore stesso. Quando si lavora nel World Wide Web, metodi come l'LSI non possono essere utilizzati. Dal capitolo precedente sappiamo che il potere dell'LSI deriva dall'uso della decomposizione ai valori singolari (SVD). Proprio per questo motivo, tale metodo può essere usato solo per piccole collezioni di documenti, infatti il calcolo e la memorizzazione dell'SVD della *matrice termine-documento* sono molto costosi, basta pensare che la *matrice termine-documento* ha tante colonne quanti sono i documenti della collezione considerata. Non è solo l'immensità del web a rendere inutilizzabili metodi tradizionali ed efficenti come l'LSI; il web possiede tante altre peculiarità che lo rendono una stimolante collezione di documenti da analizzare. Innanzitutto tali documenti non sono soggetti ad un processo di revisione editoriale: il web contiene documenti ridondanti, *links* interrotti ed anche documenti di qualità molto bassa. Inoltre è soggetto a frequenti aggiornamenti dove le pagine vengono continuamente modificate, aggiunte o cancellate. Ricordiamoci che gli utenti del web generalmente scrivono richieste molto corte, raramente fanno uso di *feedback* per revisionare

la ricerca e quasi sempre guardano solo i primi 10/20 documenti ritrovati. La più importante caratteristica che rende il web una particolare collezione di documenti è l'unicità della sua struttura ipertestuale. Questa struttura è sfruttata da tre dei più citati metodi per il *web information retrieval*: Pagerank, HITS e SALSA. In questo capitolo trattaremo il famosissimo metodo Pagerank [4], sviluppato da Sergey Brin e Larry Page, inventori di Google.

2.1 Definizione di Pagerank

Quando viene fatta una ricerca su Internet usando un motore di ricerca, vi è inizialmente una parte di *text processing* in cui lo scopo è trovare tutte le pagine web contenenti le parole della *query*. Molti motori di ricerca, incluso Google, continuamente fanno girare un esercito di programmi che recuperano pagine web e indici di parole di ciascun documento memorizzando le informazioni in un formato efficiente. Google rivendica l'indicizzazione di 25 milioni di pagine [1] e grossolanamente il 95% dei testi presenti nelle pagine web contengono più di 10^4 parole. Questo significa che, nella maggior parte delle ricerche, c'è un enorme numero di pagine contenenti le parole presenti nella richiesta dell'utente, quindi è necessario fornire un ordine secondo l'importanza delle pagine che collimano con la richiesta ed è necessario che le pagine più importanti compaiano in testa alla lista. L'algoritmo Google's Pagerank stabilisce l'importanza delle pagine web senza coinvolgere una valutazione fatta dall'uomo. È sicuramente impossibile definire una valida misura di importanza che sia ritenuta accettabile per tutti gli utenti di un motore di ricerca. Google utilizza il concetto di Pagerank come una misura della qualità delle pagine web basandosi sull'assunzione che il numero di *links* verso e da una pagina web dia informazioni sull'importanza di una pagina. Ordiniamo un sottoinsieme di pagine web da 1 a n ed indichiamo con i una particolare pagina. O_i denota l'insieme degli *outlinks* ovvero i *links* in uscita dalla pagina i verso altre pagine, per cui O_i è l'insieme delle pagine raggiungibili da i. Il numero di *outlinks* lo denotiamo con N_i. L'insieme degli *inlinks* rappresenta

tutte le pagine che hanno un *outlink* verso i ovvero tutte le pagine dalle quali è possibile raggiungere la pagina i e lo denotiamo con I_i.



In generale una pagina i acquista importanza all'aumentare del numero di *inlinks* posseduti, tuttavia un sistema di classificazione basato solo sul numero di *inlink* è facile da manipolare: quando si progetta una pagina web i con l'intento di avere un elevato numero di visite è sufficiente creare un gran numero di pagine che abbiamo *outlinks* verso i senza riguardo per il loro contenuto. Per impedire ció, è opportuno definire il rango della pagina i, cioè la sua importanza, in modo che se una pagina j con alto rango ha un *outlink* verso i, questo aumenterà l'importanza di i nella maniera seguente: il suo rango sarà una somma pesata dei ranghi delle pagine che contengono un *link* verso di essa. La pesatura è fatta in modo che il rango della pagina j, che punta verso i, venga diviso in ugual misura tra i suoi *outlinks*.
Tradotto in formule matematiche significa:

$$r_i = \sum_{j \in I_i} \frac{r_j}{N_j} \quad . \tag{2.1}$$

Questa definizione preliminare è ricorsiva, per cui il Pagerank non può essere calcolato direttamente, occorre usare un'iterazione di punto fisso. Si tenta di considerare un Pagerank iniziale r_0 e si itera:

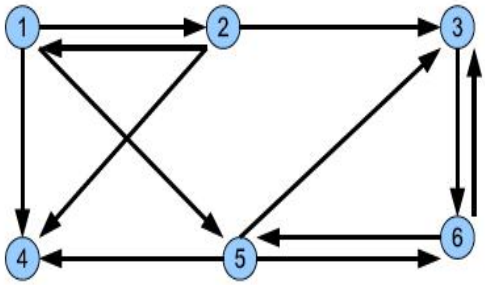
$$r_i^{(k+1)} = \sum_{j \in I_i} \frac{r_j^{(k)}}{N_j} \ , \qquad k = 0, 1, \dots \quad . \tag{2.2}$$

Ci sono alcuni problemi con questo tipo di iterazione: se una pagina non ha *outlinks* allora, nel procedimento iterativo, essa accumula rank solo attraverso gli *inlinks* quindi questo rango non viene distribuito, perciò non è chiaro se l'iterazione converge. Riformuliamo (2.1) come un problema agli autovalori per una matrice che rappresenta le adiacenze o connessioni del web cioè il grafo di Internet che stiamo considerando. Sia Q questa matrice che sarà quadrata di dimensione n definita come segue:

$$Q_{ij} = \begin{cases} \dfrac{1}{N_j} & \text{se c'è un } \textit{link} \text{ da } j \text{ a } i \text{ ,} \\ 0 & \text{altrimenti .} \end{cases}$$

Ció significa che la riga i ha elementi diversi da zero nelle posizioni che corrispondono agli *inlinks* di i e, allo stesso modo, la colonna j ha elementi diversi da zero, uguali a $\dfrac{1}{N_j}$, nelle posizioni che corrispondono agli *outlinks* di j. Si osserva che la somma degli elementi di ciascuna colonna è pari a 1 a meno che la pagina corrispondente a quella colonna non abbia *links*.

Esempio 2.1.1. Il seguente grafo illustra un insieme di pagine web con *outlinks* e *inlinks*:



La corrispondente matrice di adiacenza è

$$Q = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{3} & 0 \end{pmatrix} .$$

Poichè la pagina 4 non ha *outlinks* la corrispondente colonna ha tutti gli elementi uguali a zero.

Possiamo ora formare un vettore r le cui componenti sono i Pageranks cioè l'importanza o la classifica delle pagine e cosí la definizione (2.1) sarà equivalente al prodotto scalare fra la riga i e tale vettore r. La relazione puó essere scritta in forma matriciale come

$$\lambda r = Qr \quad \text{con} \quad \lambda = 1 \ . \tag{2.3}$$

In altre parole il vettore r è un autovettore della matrice Q corrispondente all'autovalore $\lambda = 1$, questo vettore è anche detto *vettore stazionario* di Q. L'iterazione (2.2) è equivalente a

$$r^{(k+1)} = Qr^{(k)} \ , \qquad k = 0, 1, \ldots$$

a cui viene naturale associare il *metodo delle potenze*. Comunque, a questo punto, non è ancora chiaro se il Pagerank sia ben definito, cioè non sappiamo se esista veramente un autovalore massimo e unico uguale a 1. Un aiuto ci viene fornito dalla teoria delle *catene di Markov*.

2.2 Esistenza ed unicità dell'autovalore $\lambda = 1$

Il concetto di Pagerank puó essere interpretato come una *passeggiata aleatoria* (*random walk*). Supponiamo che un navigatore, che stia visitando una pagina web, scelga la pagina successiva fra gli *outlinks* presenti con uguale probabilità ovvero supponiamo che stia viaggiando da pagina a pagina scegliendo casualmente un *link* in uscita da una pagina per andare in un'altra. Il nostro surfer, in questo modo, sta compiendo una passeggiata aleatoria in cui fare un passo significa spostarsi da una pagina web ad un'altra ed in cui la probabilità di spostarsi in una determinata pagina è data dal rango di questa. Una passeggiata aleatoria è un esempio di *catena di Markov*. Ricordiamo la definizione di quest'ultima.

Definizione 2.2.1. *Si dice catena di Markov (finita) un sistema dotato di un numero finito di stati* $\{1, 2, \ldots, n\}$ *che soddisfi la seguente ipotesi: la probabilità che il sistema passi dallo stato* i *allo stato* j *è* p_{ij}*. La matrice* $P = (p_{ij})_{i,j\in\{1,2,\ldots,n\}}$ *è detta matrice di transizione.*

Nel nostro caso, la matrice di transizione della catena non è altro che la trasposta della matrice di adiacenza ovvero Q^T.
Ritornando al nostro navigatore osserviamo che, nella sua passeggiata, egli non dovrebbe mai essere bloccato ovvero dovrebbe sempre essere libero di spostarsi da una pagina ad un'altra. In altre parole, il nostro modello di *random walk* non dovrebbe avere pagine senza *outlinks* (una tale pagina corrisponde ad una colonna di zeri in Q), quindi la matrice di adiacenza va modificata in modo che tutte le colonne di zeri siano rimpiazzate con colonne costituite da un valore costante in tutte le posizioni. Apportando questa modifica introduciamo, nella pagina che era priva di *outlinks*, un'uguale probabilità di andare in una qualsiasi pagina web. Definiamo i vettori

$$d_j = \begin{cases} 1 & \text{se } N_j = 0 \\ 0 & \text{altrimenti} \end{cases}$$

per $i = 1, \ldots, n$ ed

$$e = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n \ .$$

La matrice di adiacenza viene così modificata:

$$P = Q + \frac{1}{n} e d^T \ . \tag{2.4}$$

La matrice P ottenuta è una matrice *stocastica*: ha elementi non negativi e la somma degli elementi di ciascuna colonna è pari a 1. Le matrici stocastiche hanno numerose proprietà, due delle quali hanno per noi una particolare importanza:

- L'autovalore dominante di ogni matrice stocastica P è $\lambda = 1$.

- Una matrice stocastica P soddisfa $e^T P = e^T$.

Quindi, rendendo stocastica la matrice di adiacenza, è garantita sia l'esistenza dell'autovalore (dominante) $\lambda = 1$ sia l'esistenza di un vettore stazionario.

Esempio 2.2.1. La matrice di adiacenza Q dell'esempio precedente possiede una colonna di tutti zeri, la colonna 4, e quindi viene così modificata:

$$P = \begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{6} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{6} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{6} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{6} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{6} & \frac{1}{3} & 0 \end{pmatrix} .$$

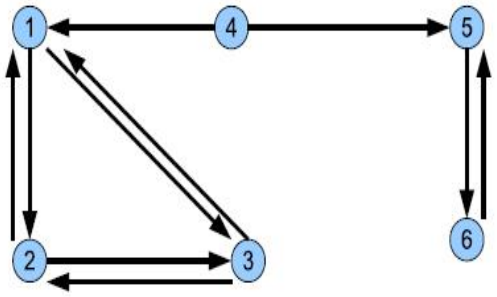
In analogia con la formula (2.3), noi vorremmo definire il vettore Pagerank come un unico autovettore di P con autovalore 1: $Pr = r$.
Tale autovettore r della matrice di transizione corrisponde alla distribuzione di probabilità stazionaria per la catena di Markov. L'elemento nella posizione i, r_i, è la probabilità che dopo un grande numero di passi il *random walker* sia alla pagina web i. Comunque l'esistenza di un unico autovalore $\lambda = 1$ non è ancora garantita. Per avere l'unicità la matrice deve essere *irriducibile*.

Definizione 2.2.2. *Una matrice quadrata A è detta riducibile se esiste una matrice di permutazione P tale che*

$$PAP^T = \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix} \tag{2.5}$$

dove X e Z sono entrambe quadrate. In caso contrario la matrice si dice irriducibile.

Esempio 2.2.2. Per illustrare il concetto di riducibilià, diamo un esempio di un grafo che corrisponde ad una matrice *riducibile*:



Un *random walker* che entra nella parte sinistra del grafo non esce più di lì e lo stesso accade se entra nella parte destra. La matrice corrispondente è

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} ,$$

che è della forma (2.5). Determinando gli autovalori si scopre che questa matrice ne ha ben due uguali a 1 ed un terzo uguale a -1.

Il grafo corrispondente ad una matrice irriducibile è detto *fortemente connesso*: dati qualsiasi due nodi (N_i, N_j) nel grafo, esiste un cammino orientato da N_i a N_j ed uno da N_j a N_i. L'unicità dell'autovalore dominante $\lambda = 1$ per una matrice stocastica ed irriducibile è una diretta conseguenza del *teorema di Perron-Frobenius*.

Teorema 2.2.1 (Perron-Frobenius)**.** *Per una matrice $A > 0$ di ordine $n \times n$ valgono le seguenti affermazioni:*

- $\exists$ *un numero positivo λ_1 autovalore di A e ogni altro autovalore λ_i (reale o complesso) in modulo è minore di $\lambda_1 : |\lambda_i| < \lambda_1$.*
- $\exists$ *un autovettore ν di A corrispondente all'autovalore λ_1 tale che tutte le sue componenti risultano positive:*

$$A\nu = \lambda_1 \nu \quad , \quad \omega^T A = \lambda_1 \omega^T \quad ; \quad \forall \ i \ \ \nu_i, \ \omega_i > 0 \ .$$

- *Non esistono altri autovettori positivi se non i multipli di ν (o di ω), tutti gli altri autovettori hanno una o più componenti complesse o negative.*

Applicando il teorema di Perron-Frobenius al nostro particolare caso in cui la matrice A è stocastica ed irriducibile, otteniamo il seguente teorema:

Teorema 2.2.2. *Sia A una matrice stocastica e irriducibile allora ha l'autovalore dominante $\lambda_1 = 1$. Esiste il corrispondente autovettore r tale che $r > 0$ e $||r||_1 = 1$ e questo è l'unico autovettore non negativo. Se $A > 0$ allora $|\lambda_i| < 1, \quad i = 2, 3, \ldots, n$.*

(La dimostrazione si trova in *C.D. Meyer Analysis and Applied Linear Algebra, SIAM 2000*).
Data la mole notevole del web possiamo stare certi che la matrice P risulta riducibile, per cui il Pagerank non è ben definito. Per assicurare l'irriducibilità, cioè per rendere impossibile che un *random walker* resti intrappolato in un sottografo, occorre aggiungere artificialmente un *link* da ogni pagina web a tutte le altre. In termini matriciali ció puó essere fatto prendendo una combinazione convessa di P e di una matrice di rango 1:

$$A = \alpha P + (1 - \alpha)\frac{1}{n}\, e\, e^T \quad , \tag{2.6}$$

per un qualche $0 \geq \alpha \geq 1$. Si verifica facilmente che A è stocastica, infatti:

$$e^T A = \alpha e^T P + (1 - \alpha)\frac{1}{n}\, e^T e\, e^T = \alpha e^T + (1 - \alpha)e^T = e^T \quad .$$

Il parametro α riveste un ruolo di primaria importanza nella computazione dell'algoritmo Pagerank di Google. Esso, da un punto di vista puramente algebrico-matematico, rappresenta la probabilità di esplorare nuove pagine web collegate tramite gli *outlinks originari* al documento elettronico correntemente visitato; tale parametro nella documentazione originale di Brian e Page risulta posto a 0.85. Contrariamente, la grandezza $1 - \alpha$ (che nel caso specifico vale 0.15) esprime la possibilità che il generico web surfer decida di intraprendere un percorso alternativo, seguendo uno dei *links creati artificialmente* e non attenendosi in questo modo al percorso indicato dai *links*

originari presenti nella pagina web correntemente visitata. A questo punto il vettore Pagerank per la matrice A risulta ben definito.

Proposizione 2.2.3. *La matrice stocastica A definita in (2.6) è irriducibile (poichè $A > 0$) ed ha autovalore dominante $\lambda_1 = 1$. Il corrispondente autovettore r è positivo, $r > 0$.*

Per la convergenza dell'algoritmo è essenziale sapere come gli autovalori di P sono cambiati tramite la combinazione convessa (2.6).

Teorema 2.2.4. *Assumiamo che gli autovalori della matrice stocastica P siano $\{1, \lambda_2, \ldots, \lambda_n\}$, allora gli autovalori di $A = \alpha P + (1-\alpha)\dfrac{1}{n} e\, e^T$ sono $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$.*

Dimostrazione. Sia $\hat{e}$ il vettore e normalizzato (con misura Euclidea pari a 1) e sia $U_1 \in R^{n\times(n-1)}$ tale che $U = (\hat{e} \quad U_1)$ sia ortogonale. Allora, siccome $\hat{e}^T P = \hat{e}^T$,

$$U^T P U = \begin{pmatrix} \hat{e}^T P \\ U_1^T P \end{pmatrix} (\hat{e} \quad U_1) = \begin{pmatrix} \hat{e}^T \\ U_1^T P \end{pmatrix} (\hat{e} \quad U_1) = \begin{pmatrix} \hat{e}^T\hat{e} & \hat{e}^T U_1 \\ U_1^T P\hat{e} & U_1^T P^T U_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \omega & T \end{pmatrix} ,$$

dove $\omega = U_1^T P\hat{e}$ mentre $T = U_1^T P^T U_1$. Notiamo che $\hat{e}\, U_1 = 0$ perché U è ortogonale. Poichè abbiamo fatto una trasformazione di similitudine, la matrice T ha autovalori $\{\lambda_2, \lambda_3, \ldots, \lambda_n\}$. Definiamo $v = \dfrac{1}{n} e$. Allora abbiamo:

$$U^T v = \begin{pmatrix} \hat{e}^T \\ U_1^T \end{pmatrix} v = \begin{pmatrix} \frac{1}{\sqrt{n}}\, e^T v \\ U_1^T v \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{n}} \\ U_1^T v \end{pmatrix} .$$

Perció,

$$U^T A U = U^T(\alpha P + (1-\alpha) v e^T) U = \alpha \begin{pmatrix} 1 & 0 \\ \omega & T \end{pmatrix} + (1-\alpha) \begin{pmatrix} \frac{1}{\sqrt{n}} \\ U_1^T v \end{pmatrix} (\sqrt{n} \quad 0) =$$

$$= \alpha \begin{pmatrix} 1 & 0 \\ \omega & T \end{pmatrix} + (1-\alpha) \begin{pmatrix} 1 & 0 \\ \sqrt{n}\, U_1^T v & 0 \end{pmatrix} =: \begin{pmatrix} 1 & 0 \\ \omega_1 & \alpha T \end{pmatrix} .$$

Da cui segue l'affermazione del teorema. □

Questo teorema implica che anche se P ha un autovalore multiplo uguale a

1, il secondo autovalore di A più grande in modulo è sempre uguale ad α. Questo è proprio ciò che accade in Internet dove la struttura del web forza λ_2 ad essere uguale ad 1 con alta probabilità. Ricordiamoci che il tasso di convergenza del metodo delle potenze è governato dalla differenza fra l'autovalore dominante e quello sottodominante, quindi scegliendo α molto minore di 1 aumenterà la differenza fra l'autovalore dominante (1) e l'autovalore sottodominante (α) e ciò velocizzerà l'algoritmo. In altre parole, gli ingegneri di Google possono stabilire il tasso di convergenza in base a quanto piccolo scelgono α. Tuttavia tale scelta non è del tutto libera da vincoli in quanto è necessario attuare un delicato bilanciamento: al diminuire di α aumenterà la velocità di convergenza ma, di conseguenza, la struttura originaria del web sarà meno usata per determinare l'importanza delle pagine. Piccole differenze nei valori di α possono produrre Pageranks molto diversi.
Negli anni Google ha apportato delle modifiche alla combinazione convessa (2.6) definendo una piú realistica e meno democratica matrice A:

$$A = \alpha P + (1 - \alpha) v\, e^T \ , \tag{2.7}$$

dove v è un vettore non negativo, con $||v||_1 = 1$, che puó essere scelto per indirizzare la ricerca verso un certo tipo di pagine web. Tale vettore è chiamato *personalization vector*. Da un punto di vista commerciale, l'aggiunta della matrice $(1 - \alpha) v\, e^T$ al posto della matrice $(1 - \alpha)\, \dfrac{ee^T}{n}$ permette di regolare piú in alto o piú in basso i valori dei Pageranks a seconda delle esigenze.
Un'ultima questione che non va dimenticata riguarda l'accuratezza dei Pageranks calcolati. Poichè r è un vettore di probabilità, ciascun r_i sarà compreso tra 0 e 1. Supponiamo che r abbia 4 miliardi di componenti. È possibile che una piccola parte della coda di questo vettore possa assomigliare a:

$$r = (\ldots, 0.000001532 \quad 0.0000015316 \quad 0.0000015312 \quad 0.0000015210, \ldots) \ .$$

Una precisione almeno dell'ordine di 10^{-9} è necessaria per distinguere gli elementi di questo sottovettore. Comunque i confronti sono fatti solo fra gli elementi di un sottoinsieme del vettore r infatti, mentre gli elementi dell'intero vettore Pagerank possono essere raggruppati in modo fitto in una parte

dell'intervallo (0,1), gli elementi del sottoinsieme relativo ad una particolare *query* sono raggruppati in modo molto meno denso, quindi è molto probabile che una precisione dell'ordine di 10^{-12} non sia necessaria per questa applicazione. Il fatto che Brin e Page riportino ragionevoli stime per r dopo solo 50 iterazioni del metodo delle potenze su una matrice dell'ordine di 322000000 ha due possibili implicazioni: o le loro stime per r non sono cosí accurate o l'autovalore sottodominante della matrice A è molto minore di 1. La prima affermazione non puó essere verificata dato che Google non ha mai pubblicato informazioni sui test di convergenza. La seconda possibilità ci dice che la matrice $\frac{e\,e^T}{n}$ (o più in generale $v\,e^T$) deve portare una buona quantità di peso e forse α è diminuito a 0.8 per aumentare la differenza fra gli autovalori e quindi la velocità di convergenza. Diminuendo il valore di α ed allo stesso tempo aumentando il peso della matrice $\frac{ee^T}{n}$ (o di $v\,e^T$), la matrice di transizione si allontana dall'originale struttura del web.
Diverse osservazioni suggeriscono che la naturale struttura del web tende a produrre una catena di Markov quasi completamente decomponibile (NCD)[1] o una catena di Markov con sottografi NCD. Se venisse scoperto che tali catene di Markov sono realmente NCD, si aprirebbe una nuova strada per la ricerca nel campo dell'*information retrieval*.

2.3 Algoritmi

Il Pagerank è solo una parte del sistema di ranking di Google, infatti è combinato con altri punteggi al fine di ottenere un ranking completo e globale. Per semplificare gli esempi presentiamo un modello base per l'uso del Pagerank costituito da due passi principali: nel primo passo si determina il sottoinsie-

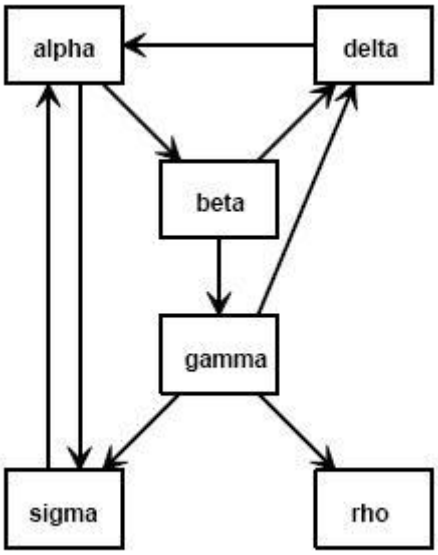
[1]Una catena di Markov è NCD se lo spazio degli stati può essere suddiviso in sottoinsiemi disgiunti con forti interazioni fra gli stati di un sottoinsieme ma con deboli interazioni fra i sottoinsiemi stessi. La matrice di transizione di una catena NCD può essere riordinata in modo da essere una matrice a blocchi dove i blocchi sulla diagonale sono densi mentre quelli fuori dalle diagonali sono sparsi.

me di nodi che contengono i termini della *query*, chiamato *sottoinsieme di rilevanza per la* query; nel secondo passo l'insieme di rilevanza è ordinato secondo i valori dei Pageranks di ciascun documento dell'insieme, quindi osserviamo che il vettore Pagerank non dipende dalla *query*. Il calcolo di tale vettore è costoso e richiede molto tempo in quanto bisogna trovare il vettore stazionario di una matrice stocastica ed irriducibile le cui dimensioni sono dell'ordine di miliardi ed il metodo delle potenze sembra essere il metodo scelto da Google. L'algoritmo per calcolare il vettore Pagerank r per la matrice di Google $A = \alpha P + (1-\alpha)\frac{ee^T}{n}$ presenta diverse varianti. Innanzitutto osserviamo che, nella pratica, la matrice di adiacenza non viene quasi mai utilizzata. Ad essa si preferisce la matrice di connettivitá G cosí definita:

$$g_{i,j} = \begin{cases} 1 & \text{se c'è un } \textit{link} \text{ da pagina } j \text{ a pagina } i\ , \\ 0 & \text{altrimenti} \quad . \end{cases}$$

È evidente che è più facile costruire questa matrice rispetto a quella di adiacenza. La matrice G sarà una matrice sparsa ovvero una matrice i cui elementi saranno soprattutto zeri.

Esempio 2.3.1. Vediamo come costruire in Matlab la matrice di connettivitá relativa alla seguente porzione di web:



Possiamo generare la matrice specificando la coppia di indici (i, j) di elementi non zero. Poichè c'è un *link* da alpha.com a beta.com l'elemento (2, 1) di G è non zero. Procedendo allo stesso modo con le altre connessioni i due vettori i e j avranno la seguente forma:

```
i = [ 2 6 3 4 4 5 6 1 1];
j = [ 1 1 2 2 3 3 3 4 6];
```

In questo caso la nostra matrice 6×6 ha 27 zeri e solo 9 elementi diversi da zero. L'istruzione

```
n = 6;
G = sparse(i,j,1,n,n);
```

genera una rappresentazione sparsa di una matrice $n \times n$ con 1 nelle posizioni specificate dai vettori i e j.

Partendo dalla matrice di connettività G, il modo migliore per scrivere in Matlab la matrice di Google A è il seguente:

$$A = pGD + ez^T \quad , \tag{2.8}$$

dove D è la seguente matrice diagonale di elementi

$$d_{j,j} = \begin{cases} \dfrac{1}{c_j} & \text{se } c_j \neq 0 \\ 0 & \text{se } c_j = 0 \end{cases} \quad ,$$

con c vettore riga le cui componenti sono la somma degli elementi di G per colonne, e è il vettore colonna di tutti 1, $p = \alpha = 0.85$ e z è il vettore di componenti

$$z_j = \begin{cases} \dfrac{(1-p)}{n} & \text{se } c_j \neq 0 \\ \dfrac{1}{n} & \text{se } c_j = 0 \end{cases} \quad .$$

Precisiamo che le due espressioni (2.6) e (2.8) sono equivalenti, cioè la matrice che viene creata è la stessa. A questo punto possiamo applicare il metodo

delle potenze: definiamo un iterato iniziale $r^{(0)} = \frac{e}{n}$, dove n è la dimensione della matrice G, e iteriamo

$$r^{(k+1)} = Ar^{(k)} \quad (k \geq 0)$$

finchè non si raggiunge il grado di convergenza desiderato, ovvero finchè la norma della differenza fra gli ultimi iterati generati, $r^{(k+1)}$ ed $r^{(k)}$, non è minore di una tolleranza fissata (per esempio 10^{-2}). Ovviamente non si memorizzano tutti gli iterati $r^{(0)}, r^{(1)}, r^{(2)}, \ldots$ ma ad ogni passo si memorizzano solo gli ultimi due iterati generati, $r^{(k-1)}$ ed $r^{(k)}$, ed al passo successivo si sovrascrivono con $r^{(k)}$ ed $r^{(k+1)}$. Tuttavia applicare il metodo delle potenze in modo convenzionale come abbiamo fatto ora non è la scelta migliore in quanto la creazione della matrice A ed il prodotto matrice-vettore $Ar^{(k)}$, che va calcolato ad ogni passo, sono molto costosi. Il metodo delle potenze può anche essere implementato in modo da non costruire la matrice di Google A. Prima di tutto si modifica la matrice di connettività G in questo modo:

$$G = p\,G\,D \ ,$$

poi si considera l'iterato iniziale $r^{(0)} = \frac{e}{n}$ e si ripete l'istruzione

$$r^{(k+1)} = Gr^{(k)} + e(zr^{(k)}) \quad (k \geq 0)$$

finchè la norma della differenza fra gli iterati $r^{(k+1)}$ ed $r^{(k)}$ non è minore di una tolleranza fissata. Questa implementazione ha due aspetti positivi: si preserva la sparsità delle matrici e la moltiplicazione matrice-vettore $Gr^{(k)}$ richiede solo prodotti scalari sparsi quindi è poco costosa. Questi prodotti scalari sparsi possono facilmente essere implementati in parallelo e l'uso del calcolo parallelo è imperativo per problemi di queste dimensioni. Ci sono stati recenti progressi nel calcolo e nell'implementazione del Pagerank, proposti per la maggior parte da ricercatori di Stanford. Arasu e alunni suggeriscono l'uso del metodo di Gauss-Seidel al posto del semplice metodo delle potenze, mentre Kamvar e alunni hanno sviluppato diverse modifiche al metodo delle potenze per accelerare la convergenza. Una tecnica usa l'estrapolazione quadratica per velocizzare la convergenza al vettore Pagerank.

Il modo migliore di calcolare il vettore Pagerank in Matlab è considerare la particolare struttura della matrice di Google A [7]. L'equazione $r = Ar$ puó essere scritta come:

$$(I - pGD)r = \gamma e \quad \text{dove} \quad \gamma = z^T r \ .$$

Il valore di γ non si conosce in quanto dipende dal vettore r che è sconosciuto ma possiamo considerare $\gamma = 1$. Se p è strettamente minore di 1 la matrice dei coefficienti $I - pGD$ è non singolare e l'equazione

$$(I - pGD)r = e$$

può essere risolta in r. In conclusione il vettore Pagerank è dato dalla soluzione, tramite l'eliminazione di Gauss, di un sistema lineare sparso e può essere scalato in modo tale che $\sum_i r_i = 1$. Notiamo che in questa implementazione il vettore z non è minimamente coinvolto.
È possibile usare anche un algoritmo chiamato *iterazione inversa* [7]. Per prima cosa si costruisce la matrice di Google in questo modo:

$$A = p\, G\, D + \frac{1-p}{n}$$

e poi si risolve il sistema lineare

$$(I - A)r = e \ \ .$$

Ad una prima occhiata questa idea potrebbe sembrare pericolosa poichè la matrice $(I - A)$ è in linea teorica singolare, ma grazie agli errori di arrotondamento è molto probabile che la matrice calcolata non lo sia, infatti, anche se inizialmente la matrice è singolare, gli errori di arrotondamento che intervengono nell'eliminazione di Gauss fanno sì che gli elementi diagonali siano difficilmente degli zeri esatti. Inoltre l'algoritmo di eliminazione di Gauss con pivoting per colonne produce una soluzione con un residuo piccolo anche se la matrice risulta mal condizionata. Il vettore r ottenuto con l'operazione

```
(I-A)\e
```

generalmente ha componenti grandi e se viene scalato in modo tale che $\sum_i r_i = 1$ il residuo risulta scalato del medesimo fattore e diventa molto piccolo. Di conseguenza, i due vettori x e Ax risultano uguali a meno di un errore di arrotondamento.
Ora applichiamo questi quattro algoritmi al grafo dell'esempio (2.3.1) e confrontiamo i quattro vettori Pagerank ottenuti.

```
-----POTENZE CONVENZIONALE-----
     page-rank  in  out  url
   1   0.3177    2    2  alpha
   6   0.2018    2    1  sigma
   2   0.1702    1    2  beta
   4   0.1383    2    1  delta
   3   0.1067    1    3  gamma
   5   0.0652    1    0  rho
----POTENZE MATRICI SPARSE----
     page-rank  in  out  url
   1   0.3177    2    2  alpha
   6   0.2018    2    1  sigma
   2   0.1702    1    2  beta
   4   0.1383    2    1  delta
   3   0.1067    1    3  gamma
   5   0.0652    1    0  rho
-------SISTEMA LINEARE-------
     page-rank  in  out  url
   1   0.3210    2    2  alpha
   6   0.2007    2    1  sigma
   2   0.1705    1    2  beta
   4   0.1368    2    1  delta
   3   0.1066    1    3  gamma
   5   0.0643    1    0  rho
------ITERAZIONE INVERSA------
```

	page-rank	in	out	url
1	0.3210	2	2	alpha
6	0.2007	2	1	sigma
2	0.1705	1	2	beta
4	0.1368	2	1	delta
3	0.1066	1	3	gamma
5	0.0643	1	0	rho

Con tutti e quattro gli algoritmi l'ordine delle pagine riportate in base alla loro importanza è lo stesso. I vettori Pagerank ottenuti sono leggermente diversi: quelli ottenuti tramite i primi due algoritmi (nei quali si usa il metodo delle potenze) sono uguali, come del resto lo sono i vettori ottenuti con gli ultimi due algoritmi (nei quali si risolve un sistema lineare). Questo ci fa capire che, in base alla situazione in cui ci troviamo, è più conveniente usare un algoritmo al posto di un altro. Come abbiamo già detto, quando si lavora in Matlab il terzo algoritmo è il migliore ma in generale, quando si lavora con vere porzioni di web le cui dimensioni sono dell'ordine di milioni di pagine, è più conveniente usare il metodo delle potenze che preserva la sparsità e non va a calcolare esplicitamente la matrice A.
Nelle tabelle, oltre ad essere riportati i valori dei Pageranks, sono anche riportati gli *url*, ovvero gli indirizzi delle pagine web, ed il numero di *inlinks* ed *outlinks* che ogni pagina ha.

2.4 Aggiornamenti

Il funzionameto dell'algoritmo Google's Pagerank oggi è ben conosciuto, tant'è che esistono diversi siti che insegnano come migliorare il Pagerank del proprio sito o più precisamente della *home page* dei quest'ultimo. A mio parere, il sito meglio organizzato è: *http://www.googlerank.it*. Esso dá informazioni su come aumentare il Pagerank e la Link Popularity, su come avere più accessi al sito e su come migliorare il posizionamento sia su Google che su altri motori di ricerca.

Per quanto riguarda l'aumento del Pagerank del proprio sito gli accorgimenti fondamentali da seguire sono:

- Cercare di ottenere piú *links* possibili verso il proprio sito. Cercare di individuare le pagine che hanno un buon Pagerank e fare attenzione nel distinguere tra il Pagerank del sito e quello della specifica pagina.
- Scegliere pagine con meno *link* possibili e controllare se la rispettiva pagina è bene indicizzata da Google.
- Organizzare bene il sito, ovvero controllare la quantità di *link* esterni ed interni. Evitare di inserire troppi outbound *links* (verso altri siti) dalle pagine con un PR alto, perché un'importante fetta del PR lo si trasmette fuori invece di trasferirlo alle proprie pagine interne. Adottare la via tradizionale, di creare una pagina dedicata ai *links* esterni, e collegare gli altri siti da quella. Inoltre è tecnica diffusa la ricerca del massimo PR per la *home page*. Fare quindi un *link* diretto a questa da tutte le altre pagine.
- Evitare i contenuti duplicati. Nell'algoritmo del Pagerank, al contrario di altri motori che non conoscono ancora il concetto di contenuto duplicato, Google riesce ad individuare le pagine copiate.

È possibile conoscere il valore del Pagerank di una specifica pagina tramite la Google ToolBar che assegna valori tra 0 e 10 calcolati sulla base di una scala logaritmica. Perció, la Google ToolBar non mostra il reale valore del PR (conosciuto solo da Google stesso) ma il range nel quale questo è posizionato. Ecco un esempio che visualizza i valori reali e i valori della Toolbar:

PR reale	PR sul Toolbar	
1-10	1	Quasi tutte le nuove pagine
10-100	2	Pagine con qualche *link* esterno
100-1000	3	Inizio della diffusione
1.000-1.0000	4	Popolaritá media

10.000-100.000	5	Popolaritá media, *linking* interno ottimizzato
100.000-1.000.000	6	Pagina di un sito importante
1.000.000 - 10.000.000	7	Irraggiungibile per il normale webmaster

Questo spiega che stessi valori visualizzati sulla Toolbar possono rappresentare PR reali molto diversi. Ed è per questo che il passaggio da PR 4 a 5 è abbastanza facile, ma da 5 a 6 diventa difficile. Enècessario diventare dieci volte piú famosi per riuscirci! Il PR reale ovviamente non è un semplice calcolo del numero dei *links*. Contano i siti che ti *linkano*, il testo con cui ti *linkano*, la rilevanza della tua pagina rispetto al nome del *link*, l'ottimizzazione del *linking* interno, il contenuto e a volte anche il successo della pagina stessa: Google riesce a capire se una volta arrivato su una pagina tramite un *link*, il visitatore è annoiato e se ne va via o se ha trovato veramente quello che voleva e la sua visita si conclude con una conversione.
Attualmente il Pagerank non è usato direttamente nell'algoritmo di posizionamento. Questo può essere anche ovvio, in quanto il PR puro caratterizza soltanto il numero di *links* qualitativi al sito, ma ignora completamente il testo dei *links* e il contenuto delle pagine di provenienza. Questi fattori sono importanti nel PR, e sono stati inseriti nell'algoritmo successivamente.
Google in due anni ha creato ben due nuovi algoritmi. Essi si chiamano Panda, uscito il 12 agosto 2011, e Penguin, uscito nell' aprile 2012.
Mentre Panda si occupa di selezionare i contenuti degli articoli e di scartare quelli con poco testo in favore dei piú elaborati, Penguin si occupa delle parole chiave usate, utile mezzo per indicizzare i nostri articoli sui motori di ricerca. In entrambi i casi lo scopo è: eliminare lo spam e le copie di altri articoli per diminuire il piú possibile il carico di informazioni 'fantasma' (cioè senza contenuto originale). Lo stesso Google, il 18 maggio 2011, ha rilasciato una serie di domande generali per capire se un nostro articolo è ben costruito. Non dá risposte, ma i quesiti ci permettono indirettamente di farci un'idea su dove è necessario migliorare. Da queste domande si evince che l'unica vera regola è: *'Costruire un articolo dai contenuti unici, possibilmente ben*

descritti, con più di qualche frase introduttiva'. Se gli spider di Google (robot che setacciano i siti in cerca di contenuti) trovano anche un solo articolo chiaramente poco adatto, l'intero vostro sito potrebbe essere penalizzato.
In linea generale:

- Non copiare pagine dall'esterno, ma prendere solo parte dei contenuti ed elaborarli. Meglio ancora se producete materiale solo vostro.
- Non usate il vostro sito come semplice aggregatore di *link*. Mettere dei contenuti è imperativo.
- Evitate di inserire troppa pubblicitá.
- Se gran parte degli utenti abbandonano il vostro sito dopo aver visitato una sola pagina, Google potrebbe scegliere di valutarla poco. Cercate quindi un sistema per tenere ancorati gli utenti al vostro sito.
- Stessa cosa del punto precedente vale per gli utenti di ritorno. Se Google si accorge che gli utenti visitano il sito e non tornano in futuro, riterrá il vostro sito poco interessante.

A distanza di meno di un anno, fá la sua comparsa l'algoritmo Penguin, attivo dal 24 aprile 2012. Il suo scopo è ben diverso dal precedente aggiornamento: impedire il *keyword stuffing*, cioè l'uso sconsiderato delle parole chiave al fine di promuovere il proprio sito. Nella pratica, chi usa questa tecnica inserisce parole chiave che non hanno niente a che fare con l'argomento della pagina (in genere sono legate a temi come il porno o il gossip). Penguin si occupa di penalizzare questo tipo di siti e non solo, infatti se voi puntate con un *link* al sito penalizzato in questione, molto probabilmente anche il vostro sito sará penalizzato. Google, in questo modo, spinge ad avere piú coscienza dei contenuti.

Capitolo 3

HITS e SALSA

Algoritmi quali l'algoritmo HITS (*Hipertext Induced Topic Search*) di Kleinberg [5], [6], il Pagerank di Brin e Page e l'algoritmo SALSA (*Stochastic Approach for Link Structure Analysis*) di Lempel e Moran [6] usano la struttura ipertestuale di un network di pagine web per assegnare pesi a ciascuna delle pagine. Tali pesi possono essere usati per classificare le pagine in base alla loro importanza ed autorevolezza. Questi tre algoritmi hanno un aspetto in comune: trovano un autovettore dominante di una matrice non negativa che descrive la struttura ipertestuale del dato network e usano le componenti di questo autovettore come pesi per le pagine. Andiamo adesso ad analizzare uno per volta questi due nuovi algoritmi introdotti.

3.1 HITS

Ricordiamo che ciascuna pagina (documento) del web è rappresentabile come nodo di un grafo molto grande. Gli archi che collegano questi nodi rappresentano i *links* tra i documenti. Il metodo HITS definisce *authorities* (pagine autorevoli) e *hubs* (pagine rilevanti): un'*authority* è un documento con diversi *inlinks* mentre un *hub* ha diversi *outlinks*. La tesi di HITS è che buoni *hubs* puntano a buone *authorities* e buone *authorities* sono puntate da buoni *hubs*. HITS assegna sia un punteggio (*score*) come *hub* sia un punteggio come *au-*

thority a ciascuna pagina web, quindi la generica pagina i ha sia un'*authority score* x_i sia un *hub score* y_i. Indichiamo con E l'insieme di tutti i collegamenti diretti nel grafo che rappresenta il web e chiamiamo e_{ij} il collegamento, ovvero il *link*, dal nodo i al nodo j. A ciascuna pagina sono stati assegnati un iniziale *authority score* $x_i^{(0)}$ e *hub score* $y_i^{(0)}$ e HITS successivamente raffina questi punteggi calcolando:

$$x_i^{(k)} = \sum_{j:\, e_{ij} \in E} y_j^{(k-1)} \quad \text{e} \quad y_i^{(k)} = \sum_{j:\, e_{ij} \in E} x_j^{(k)} \quad \text{per } k = 1, 2, 3, \ldots \ . \tag{3.1}$$

Queste equazioni possono essere scritte in forma matriciale con l'aiuto della *matrice di adiacenza* L che rappresenta il grafo.

$$L_{i,j} = \begin{cases} 1 & \text{se c'è un link da pagina } i \text{ a pagina } j \ , \\ 0 & \text{altrimenti} \quad . \end{cases}$$

Notiamo che questa definizione di matrice di adiacenza è diversa da quella incontrata nel precedente capitolo (questa nuova matrice non è altro che la trasposta della matrice di connettività vista precedentemente) ma, poiché la letteratura inerente all'algoritmo HITS chiama matrice di adiacenza la matrice appena definita, ho preferito attenermi a tale convenzione.
In notazioni matriciali, l'equazione (3.1) assume la seguente forma:

$$x^{(k)} = L^T y^{(k-1)} \quad \text{e} \quad y^{(k)} = L^T x^{(k)} \ .$$

Questo porta al seguente algoritmo per calcolare l'*authority score* x e l'*hub score* y.

1. Inizializzazione: $y^{(0)} = e$ dove e è un vettore colonna di tutti 1. Altri vettori iniziali positivi possono essere usati.

2. Fino a convergenza ripetere:

$$x^{(k)} = L^T y^{(k-1)}$$
$$y^{(k)} = L^T x^{(k)}$$
$$k = k + 1$$
$$Normalizzare \ \ x^{(k)} \ \ e \ \ y^{(k)} \ .$$

Nel passo 2 dell'algoritmo, notiamo che le due equazioni $x^{(k)} = L^T y^{(k-1)}$ e $y^{(k)} = L^T x^{(k)}$ possono essere semplificate per sostituzione e quindi diventano

$$x^{(k)} = L^T L x^{(k-1)} \quad \text{e} \quad y^{(k)} = L L^T y^{(k-1)} \ .$$

Queste due equazioni definiscono il *metodo delle potenze* per calcolare l'autovettore dominante delle matrici $L^T L$ e LL^T. Poiché la matrice $L^T L$ determina l'*authority score*, essa è chiamata *matrice authority*, mentre la matrice LL^T è conosciuta come *matrice hub*. Notiamo che $L^T L$ e LL^T sono matrici simmetriche semidefinite positive. In conclusione, calcolare il vettore *authority* x ed il vettore *hub* y significa trovare l'autovettore dominante di $L^T L$ e di LL^T rispettivamente.
L'implementazione di HITS include due passi principali. Per prima cosa è creato un *grafo delle vicinanze* N (*neighborhood graph* N) collegato ai termini della *query*. Successivamente, sono calcolati l'*authority score* e l'*hub score* per ciascun documento in N e due liste ordinate, contenenti una i documenti più autorevoli e l'altra i più '*hubby*' (rilevanti), sono presentate all'utente. Poiché abbiamo giá trattato del secondo passo, concentriamoci sul primo. Tutti i documenti che contengono riferimenti ai termini della *query* sono posizionati nel grafo delle vicinanze N. Ci sono vari modi per determinare questi documenti ed uno dei più semplici si basa sul *file* termine-documento invertito. Supponiamo che il *file* abbia la seguente forma:

- aardvark: termine 1 - doc 3, doc 117, doc 3961
- $\vdots$
- aztec: termine 10 - doc 15, doc 3, doc 101, doc 19, doc 1199, doc 673
- baby: termine 111 - doc 56, doc 94, doc 31, doc 3
- $\vdots$
- zymurgy: termine m - doc 223

Per ciascun termine i documenti che menzionano tale termine sono classificati in una lista, per esempio una *query* sui termini 10 e 11 convoglierà in N i documenti 15, 3, 101, 19, 1199, 673, 56, 94 e 31. In seguito il grafo attorno al sottoinsieme di nodi in N verrà espanso aggiungendo nodi che puntano ai nodi in N o sono puntati dai nodi in N. Questa espansione permette di fare qualche associazione semantica latente, per esempio se il termine *car* compare nella *query*, attraverso l'espansione sui documenti che contengono *car* qualche documento contenente *automobile* può essere aggiunto a N. In questo modo si spera di risolvere il problema dei sinonimi. Purtroppo l'insieme N può diventare molto grande a causa di questo processo di espansione quindi, nella pratica, viene fissato il numero massimo di *inlinks* e *outlinks* da aggiungere per ogni specifico nodo in N. Una volta che il grafo N è stato costruito, viene creata la matrice di adiacenza L corrispondente a tale grafo. La dimensione di L è molto più piccola del numero totale di documenti presenti nel web, quindi calcolare l'*authority score* e l'*hub score* usando gli autovettori dominanti di L^TL e LL^T comporta un minore costo rispetto a calcolare l'*authority score* e l'*hub score* quando tutti i documenti del web sono presenti in N. Inoltre notiamo che si ha un'ulteriore riduzione del costo computazionale: è necessario calcolare solo un autovettore, quello della matrice L^TL o quello di LL^T ma non entrambi. Per esempio il vettore *authority* x può essere ottenuto calcolando l'autovettore dominante di L^TL poi il vettore *hub* y può essere ottenuto dall'equazione $y = Lx$. Un simile scenario si verifica se, dal problema agli autovalori, si calcola prima il vettore *hub*.
Come abbiamo detto, l'algoritmo iterativo per calcolare i vettori HITS è il metodo delle potenze applicato alle matrici L^TL ed LL^T. Per una matrice diagonalizzabile B di ordine $n \times n$ i cui autovalori distinti sono $\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ con $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \ldots \geq |\lambda_k|$, il metodo delle potenze prende un vettore iniziale $x^{(0)}$ e iterativamente calcola

$$x^{(k)} = Bx^{(k-1)} \quad , \quad \frac{x^{(k)}}{m(x^{(k)})} \longrightarrow x^{(k)},$$

dove $m(x^{(k)})$ è una costante di normalizzazione e solitamente coincide con la componente di $x^{(k)}$ più grande in valore assoluto e presa con il suo segno.

In questo caso $m(x^{(k)})$ converge all'autovalore dominante λ_1 e $x^{(k)}$ ad un suo autovettore normalizzato. Se ci interessa conoscere solo l'autovettore dominante (e non λ_1) può essere usata una normalizzazione quale $m(x^{(k)}) = ||x^{(k)}||$. Le matrici L^TL e LL^T sono simmetriche, semidefinite positive e non negative così i loro autovalori distinti $\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ sono necessariamente reali e non negativi con $\lambda_1 > \lambda_2 > \ldots > \lambda_k \geq 0$ per cui non è possibile avere autovalori multipli nello spettro. Di conseguenza si evitano i maggiori problemi di convergenza: l'algoritmo HITS normalizzato converge sempre.

Teorema 3.1.1. *La sequenza* $\left\{\frac{x^{(k)}}{m(x^{(k)})}\right\}_{k\geq 1}$ *ottenuta tramite l'algoritmo HITS converge ad un* authority vector x *che è un autovettore non negativo relativo all'autovalore dominante di* L^TL. *Allo stesso modo la sequenza* $\left\{\frac{y^{(k)}}{m(y^{(k)})}\right\}_{k\geq 1}$ *converge ad un* hub vector *che è un autovettore non negativo relativo all'autovalore dominante di* LL^T.

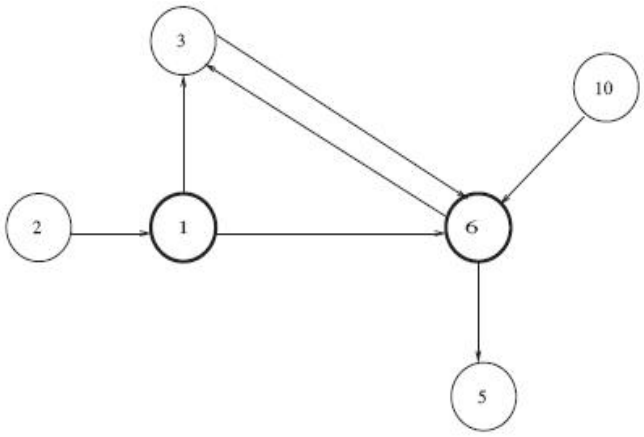
Dimostrazione. Gli autovalori di LL^T sono reali e non negativi. Segue che, mentre l'autovalore di modulo maggiore può essere multiplo, tutti gli altri autovalori hanno modulo strettamente minore. Poichè LL^T è simmetrica, gli autospazi sono ortogonali. Nell'autospazio relativo all'autovalore dominante possiamo usare l'ortonormalizzazione di Gram-Schmidt per scegliere vettori ortogonali così che uno di loro sia non negativo. Dato che $y^{(0)}$ è positivo, il prodotto scalare di questo con il vettore non negativo è positivo e così $y^{(0)}$ ha una componente non banale nell'autospazio generato dall'autovalore dominante. Questo assicura che l'algoritmo converga ad un autovettore relativo all'autovalore dominante. Lo stesso ragionamento può essere applicato al vettore *authority*. □

Notiamo che, sebbene l'algoritmo converga, esso può convergere a qualsiasi vettore normalizzato non negativo che si trova nell'autospazio generato dall'autovalore dominante λ_1 quindi ci possono essere problemi legati all'unicità dei vettori *authority* e *hub*. Se $\lambda_1 > \lambda_2$ la struttura di L può permettere che λ_1 sia una radice multipla del polinomio caratteristico ed in questo caso l'autospazio associato sarebbe multidimensionale, ciò significa che differenti

vettori *authority* (e vettori *hub*) possono essere prodotti da scelte diverse del vettore iniziale $x^{(0)}$. Al centro del problema di unicità, risiede il problema della riducibilità (Definizione (2.2.2) a pagina 27). Ricordiamo che il teorema di Perron-Frobenius assicura che una matrice irriducibile non negativa possiede un unico autovettore dominante positivo di norma 1, di conseguenza è la riducibilità della matrice $L^T L$ che porta l'algoritmo HITS a convergere a soluzioni non uniche. Anche nel caso del Pagerank si avevano gli stessi problemi di unicità che venivano risolti modificando la matrice stocastica P in modo da ottenere una matrice irriducibile. Una modifica simile a quella di Google può essere applicata alla matrice L di HITS. Un secondo problema che si ha con il metodo delle potenze riguarda il vettore iniziale $x^{(0)}$. In generale la convergenza ad un vettore diverso da 0 si ha se il vettore iniziale $x^{(0)}$ non si trova nel range di $B - \lambda_1 I$ e se $x^{(0)}$ è generato in maniera casuale si ha un'alta probabilità che questa condizione venga soddisfatta. Tuttavia, in generale, avere una forte dipendenza dal vettore iniziale crea dei problemi in quanto può capitare di ottenere risultati non intuitivi e che sono incoerenti con la richiesta fatta; in particolare può accadere che, per nodi apparentemente importanti di un certo grafo, le relative componenti dell'*authority vector* siano zero. Questo problema è chiamato *nil-weighting* (peso zero). Per risolvere questi due problemi (non unicità della soluzione e *nil-weighting*) è stata introdotta una modifica all'algoritmo HITS chiamata *Expotentiated Input to HITS*.
Prima di trattare questa variante dell'algoritmo, vediamo un piccolo esempio per mostrare l'implementazione di HITS.

Esempio 3.1.1. Innanzi tutto, un utente presenta una richiesta al sistema di *information retrieval* HITS. Ci sono diversi schemi che possono essere usati per determinare quali nodi contengono i termini della *query*. Per esempio si possono scegliere quei nodi che contengono almeno un termine della *query* o, per creare un più piccolo grafo sparso, si possono prendere quei nodi che contengono tutti i temine della *query*. Per il nostro esempio, supponiamo che il sottoinsieme di nodi contenente i termini della *query* sia $\{1, 6\}$, ovve-

ro i documenti 1 e 6 contengono i termini della richiesta. Successivamente costruiamo il *neighborhood graph* attorno ai nodi 1 e 6. Supponiamo che ciò produca il seguente grafo N.



Dato N si costruisce la matrice di adiacenza L:

$$L = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} .$$

Le matrice *authority* e *hub* sono rispettivamente:

$$L^T L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} , \quad LL^T = \begin{pmatrix} 2 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} .$$

I principali autovettori normalizzati, ovvero l'*authority score* x e l'*hub score* y, sono:

$$x^T = (0 \quad 0 \quad 0.3660 \quad 0.1340 \quad 0.5 \quad 0) , \quad y^T = (0 \quad 0 \quad 0.3660 \quad 0.1340 \quad 0.5 \quad 0) .$$

Nella pratica si lavora con matrici di dimensioni elevate e, in questo caso, è poco probabile riscontrare valori identici negli autovettori dominanti. Tuttavia possono verificarsi uguaglianze nelle componenti dei vettori e, in questo caso, si utilizza una qualsiasi strategia di spareggio. Usando la strategia *first come, first serve* (primo arrivato, prima servito), l'*authority score* e l'*hub score* sono posti in ordine decrescente e la classifica dei documenti, ciascuno rappresentato da un numero che lo identifica, è presentata:

Authority ranking = (6 3 5 1 2 10) , Hub ranking = (1 3 6 10 2 5) .

Ciò significa che il documento 6 è il documento più autorevole per la *query* mentre il documento 1 è il più attinente per questa stessa *query*.

3.2 Exponentiated Input to HITS

L'idea chiave dell' Exponentiated Input to HITS [5] è di rimpiazzare la matrice di adiacenza utilizzata nell'algoritmo con una *matrice esponenziale* che contenga informazioni dirette su paths del grafo di lunghezza 2 o maggiore. La forte dipendenza di HITS dalla matrice di adiacenza L costringe a considerare, in un dato passo iterativo, solo percorsi di lunghezza 1 e ciò significa che, per determinare l'*hub score* di una pagina, ad ogni iterazione HITS utilizza solo gli *authority score* delle pagine adiacenti. Quello che facciamo è sostituire L con una nuova matrice che abbia informazioni su *paths* di lunghezza maggiore. Per considerare *paths* di lunghezza maggiore di 1 notiamo che, data una matrice di adiacenza L, il numero di *paths* di lunghezza m che vanno dal nodo i al nodo j è dato dalla componente (i, j) della matrice L^m. Per noi i *paths* di lunghezza 1 sono più importanti di quelli di lunghezza maggiore. Tenendo conto di quanto detto, utilizziamo al posto della matrice L questa nuova matrice:

$$L + L^2/2! + L^3/3! + \ldots + L^m/m! + \ldots = e^L - I. \tag{3.2}$$

Questa serie converge poiché ogni componente del termine m-esimo della serie può essere limitata da $\dfrac{n^m}{m!}$.

Usando la nuova matrice esponenziale $e^L - I$ l'algoritmo HITS con input esponenziale può essere espresso in termini algebrici come segue. Inizializziamo i vettori *authority* e *hub* in modo che $x_i^{(0)} > 0$ e $y_i^{(0)} > 0 \ \forall i$ e $\sum_i (x_i^{(0)})^2 = \sum_i (y_i^{(0)})^2 = 1$. Sostituiamo la matrice di adiacenza L con la matrice $e^L - I$ e così definiamo:

$$x^{(k)} = \phi_k(e^L - I)y^{(k-1)} , \quad y^{(k)} = \psi_k(e^L - I)x^{(k)} \quad \text{per } k > 0 , \tag{3.3}$$

dove ϕ_k e ψ_k sono le costanti di normalizzazione. Il teorema (3.1.1) si può applicare anche alle matrici $(e^L - I)^T(e^L - I)$ e $(e^L - I)(e^L - I)^T$ per cui la sequenza $\{x^{(k)}\}_{k\geq 1}$ si avvicina all'autospazio generato dall'autovalore dominante di $(e^L - I)^T(e^L - I)$ e la sequenza $\{y^{(k)}\}_{k\geq 1}$ si avvicina all'autospazio generato dall'autovalore dominante di $(e^L - I)(e^L - I)^T$, ovvero si ha la convergenza dell'algoritmo. Ora proviamo che, dato un grafo debolmente connesso[1], questa modifica dell'algoritmo HITS con input esponenziale garantisce l'unicità dell'autovettore impedendo in questo modo il ritorno di rankings non unici che dipendono dal vettore iniziale ed il ritorno di inappropriati pesi uguali a zero (*nil-weighting*). Anche altre matrici come $L + \frac{L^2}{2}$ e $I + L$, piuttosto che $e^L - I$, possono escludere questi due problemi sempre che il grafo G sia debolmente connesso.

Teorema 3.2.1 (Exponentiated Input to HITS)**.** *Sia G un grafo diretto con matrice di adiacenza L. Se G è debolmente connesso l'*authority vector *generato dall'algoritmo HITS con input esponenziale esiste ed è unico. Inoltre tale vettore è non negativo, ha norma 1 e tutti i nodi con un numero positivo di* inlinks *ricevono pesi positivi. Le stesse affermazioni valgono anche per l'*hub vector*: se G è debolmente connesso l'*hub vector *generato dall'algoritmo HITS con input esponenziale esiste, è unico, è non negativo ed ha norma 1. Inoltre tutti i nodi con un numero positivo di* outlinks *ricevono pesi positivi.*

[1]Un grafo orientato G si dice debolmente connesso se due qualsiasi nodi N_i e N_j sono uniti da un cammino non orientato.

Prima di passare alla dimostrazione del teorema, enunciamo un lemma ed un secondo teorema che ci serviranno nella dimostrazione. Questo secondo teorema rappresenta una variante del teorema (2.2.1) di Perron-Frobenius.

Lemma 3.2.2. *Se B è una matrice quadrata reale, tutti gli autovalori di B^TB sono reali e non negativi con i relativi autospazi di rango massimo.*

Dimostrazione. Gli autovalori sono reali e non negativi perchè $M = B^TB$ è simmetrica. Per mostrare che sono non negativi dobbiamo provare che $x^TMx \geq 0$ per ogni vettore x. Si ha che: $x^TMx = x^TB^TBx = |Bx|^2 \geq 0$. □

Teorema 3.2.3. *Una matrice irriducibile e non negativa M ha sempre un autovalore positivo λ che è una radice semplice del polinomio caratteristico ed inoltre i moduli di tutti gli altri autovalori sono minori di λ. L'autovettore corrispondente a λ può essere scalato in modo che tutte le sue componenti siano positive.*

Ora siamo pronti per dimostrare il teorema Exponentiated Input to HITS.

Dimostrazione. Proviamo questo risultato mostrando che la matrice *authority* $(e^L - I)^T(e^L - I)$ ha un semplice autovalore dominante il cui autovettore ha componenti positive esattamente per quei nodi che hanno un numero positivo di *inlinks*. La prova per la matrice *hub* è analoga.
Senza perdita di generalità, assumiamo che G abbia n nodi, t dei quali non hanno *inlinks* ed indichiamo questi ultimi con $n - t + 1, \ldots, n$. La matrice di adiacenza è data da

$$L = \begin{pmatrix} \tilde{L} & 0 \\ \tilde{B} & 0 \end{pmatrix} ,$$

dove $\tilde{L}$ è di ordine $(n - t) \times (n - t)$ e la matrice $\tilde{B}$ è di ordine $t \times (n - t)$. L'ultima t-esima colonna di

$$e^L - I = L + \frac{L^2}{2!} + \ldots + \frac{L^m}{m!} + \ldots$$

ha solo componenti pari a zero, quindi

$$(e^L - I)^T(e^L - I) = \begin{pmatrix} C & 0 \\ 0 & 0 \end{pmatrix} ,$$

dove $C = (e^{\tilde{L}} - I)^T(e^{\tilde{L}} - I)$.
Inoltre il polinomio caratteristico di $(e^L - I)^T(e^L - I)$ è $\lambda^t p_C(\lambda)$ dove $p_C(\lambda)$ è il polinomio caratteristico di C. Se C ha un autovalore dominante positivo e semplice lo stesso sarà per la matrice $(e^L - I)^T(e^L - I)$. Notiamo che se il corrispondente autovettore di C è positivo, il corrispondente autovettore di $(e^L - I)^T(e^L - I)$ sarà lo stesso vettore al quale sono state aggiunte componenti pari a zero per tutti quei nodi che non hanno *inlinks*. Il lemma (3.2.2) mostra che gli autovalori di C sono reali e non negativi. Ora abbiamo bisogno di dimostrare che C è irriducibile in modo da poter applicare il teorema (3.2.3). Chiamiamo G' il grafo la cui matrice di adiacenza (pesata) è data da C e notiamo che G' può essere considerato un grafo non orientato dato che C è simmetrica. C è irriducibile se e solo se G' è connesso quindi ora cerchiamo di dimostrare che G' è connesso.
La componente (i, j) di C è diversa da zero se e solo se c'è un nodo k tale che il grafo originale G ha un *path* da k ad i ed uno da k a j. Da questo possiamo concludere che tutti i nodi che possono essere raggiunti da un nodo k seguendo i *links* in G devono trovarsi nella stessa componente di G'. Inoltre, se k possiede *inlinks*, si ha che anche k si trova in quella componente di G'. Ora assumiamo per assurdo che G' non sia connesso, quindi può essere suddiviso in due componenti H_1 e H_2 (che possono essere disconnesse al loro volta). Queste due componenti non sono collegate ovvero non ci sono *links* fra H_1 e H_2 in G. Poiché G è debolmente connesso, deve esserci un nodo senza *inlinks* che è collegato a qualche nodo l in H_1 e ad un nodo m in H_2. In questo modo si ha un connessione fra questi nodi l ed m in G' e ciò contraddice l'ipotesi che G' non sia connesso. In conclusione abbiamo dimostrato che G' è connesso e quindi C è irriducibile. Questo completa la dimostrazione. □
Un attento esame della dimostrazione mostra che se la matrice $L + \frac{L^2}{2}$, più facilmente computabile, fosse usata al posto di $e^L - I$ il risultato varrebbe ancora. Utilizzando la matrice $L + \frac{L^2}{2}$ si osserva che tutti i nodi che possono essere raggiunti da un dato nodo i seguendo i cammini in G sono ancora

nella stessa componente di G'. Ciò si ha perché se abbiamo in G un *link* da i a j e un *link* da j a k allora che vi è anche un *link* da j a k in G'. Di conseguenza la dimostrazione vista sopra rimane valida. Alternativamente, possiamo utilizzare la matrice $I + A$ dato che il grafo G' contiene il grafo G (ignorando le direzioni) come sottografo. Di conseguenza, se G è debolmente connesso, G' sarà connesso. Poichè G' è connesso, le stesse argomentazioni viste sopra mostrano che l'algoritmo converge ad un autovettore unico.

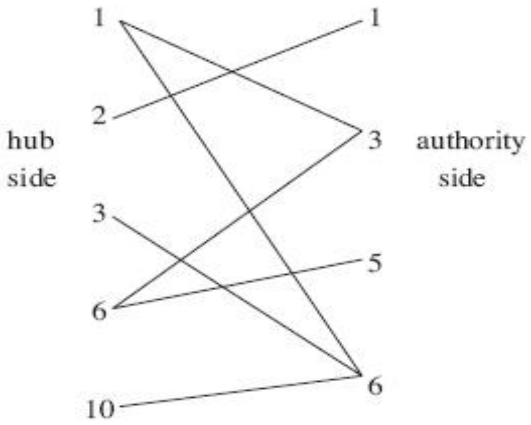
3.3 SALSA

Ora parliamo del terzo metodo per l'*information retrieval* sul web: Stochastic Approach for Link Structure Analysis (SALSA). SALSA è nato dalla combinazione delle idee di HITS e Pagerank. Come HITS, sono creati *authority* e *hub scores* per le pagine web e, come Pagerank, questi punteggi sono creati attraverso l'uso delle catene di Markov. Trattiamo il funzionamento di SALSA attraverso un esempio.

Esempio 3.3.1. In una maniera simile a HITS, viene creato il grafo delle vicinanze N associato ad una particolare richiesta. Noi usiamo lo stesso *neighborhood graph* N dell'esempio (3.1.1) precedente. SALSA differisce da HITS nel successivo passo: invece di formare una matrice di adiacenza L per il grafo delle vicinanze N, viene creato un grafo G bipartito non orientato. G è definito da tre insiemi: V_h, V_a ed E, dove V_h è l'insieme dei *nodi hub* (tutti i nodi in N che hanno un numero di *outlinks* > 0), V_a è l'insieme dei *nodi authority* (tutti i nodi in N che hanno un numero di *inlinks* > 0) ed E è l'insieme degli archi orientati in N. Notiamo che un nodo in N può essere sia nell'insieme V_h sia in V_a. Per esempio, con il grafo delle vicinanze dell'esempio (3.1.1), otteniamo i seguenti insiemi

$$V_h = \{1, 2, 3, 6, 10\} \ , \ V_a = \{1, 3, 5, 6\} \ .$$

Il grafo bipartito e non orientato G ha un *lato hub* ed un *lato authority*. I nodi in V_h sono elencati sul *lato hub* e i nodi in V_a si trovano sul *lato authority*. Ogni arco orientato in E è rappresentato da un arco non orientato in G.



Successivamente due catene di Markov sono formate partendo da G: una catena di Markov riferita ai *nodi hub* con matrice di transizione H e una catena di Markov riferita ai *nodi authority* con matrice A. Ricordiamoci che HITS calcola gli *authority* e gli *hub scores* usando la matrice non pesata L, mentre Pagerank calcola una misura analoga all'*authority score* usando una matrice pesata (per colonne) P. Per calcolare i suoi *authority* e *hub scores*, SALSA usa sia la pesatura per colonne sia la pesatura per righe. Indichiamo con L_r la matrice L in cui ciascuna riga diversa dal vettore nullo è divisa per la somma delle componenti di tale riga e indichiamo con L_c la matrice L in cui ciascuna colonna diversa dal vettore nullo è divisa per la somma delle componenti di tale colonna. A questo punto possiamo costruire H ed A: la prima sarà formata dalle righe e dalle colonne non nulle della matrice $L_r L_c^T$ mentre la seconda dalle righe e dalle colonne non nulle della matrice $L_c^T L_r$. L'approccio che abbiamo utilizzato per costruire H ed A rivela chiaramente

le connessioni che SALSA ha con HITS e Pagerank. Nel nostro esempio:

$$L_r = \begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad L_r = \begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \end{pmatrix},$$

$$L_r L_c^T = \begin{pmatrix} \frac{5}{12} & 0 & \frac{2}{12} & 0 & \frac{3}{12} & \frac{2}{12} \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 1 & 0 & \frac{3}{4} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix}, \quad L_c^T L_r = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{6} & 0 & \frac{5}{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Considerando solo le righe e le colonne non nulle di $L_r L_c^T$, otteniamo la seguente matrice H:

$$H = \begin{pmatrix} \frac{5}{12} & 0 & \frac{2}{12} & \frac{3}{12} & \frac{2}{12} \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{4} & 0 & 0 & \frac{3}{4} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} \end{pmatrix}.$$

Allo stesso modo otteniamo:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{6} & 0 & \frac{5}{6} \end{pmatrix}.$$

Se G è connesso, H ed A sono entrambe catene di Markov irriducibili e π_h^T, il vettore stazionario di H, fornisce gli *hub scores* per la *query* con grafo delle vicinanze N mentre π_a^T fornisce gli *authority scores*. Se G non è connesso, H ed A contengono componenti irriducibili multiple, ciascuna delle quali ha un proprio vettore stazionario. In questo caso, l'*hub score* e l'*authority score* globali si ottengono mettendo insieme i vettori stazionari (opportunamente pesati) relativi a ciascuna componente irriducibile individuale. Dalla struttura delle matrici H ed A del nostro esempio, è facile vedere che G non è connesso, così H ed A contengono componenti connesse multiple. In particolare H ha due componenti connesse, $C = \{2\}$ e $D = \{1, 3, 6, 10\}$, mentre le componenti connesse di A sono: $E = \{1\}$ ed $F = \{3, 5, 6\}$. Dalla struttura di H ed A emerge anche la periodicità delle catene di Markov. Tutte le componenti irriducibili di H ed A contengono cicli interni e questo implica che le catene sono aperiodiche. I vettori stazionari per le due componenti irriducibili di H sono:

$$\pi_h^T(C) = (\overset{2}{1}) , \quad \pi_h^T(D) = (\overset{1}{\frac{1}{3}} \quad \overset{3}{\frac{1}{6}} \quad \overset{6}{\frac{1}{3}} \quad \overset{10}{\frac{1}{6}}) ,$$

mentre i vettori stazionari per le due componenti irriducibili di A sono:

$$\pi_a^T(E) = (\overset{1}{1}) , \quad \pi_a^T(F) = (\overset{3}{\frac{1}{3}} \quad \overset{5}{\frac{1}{6}} \quad \overset{6}{\frac{1}{2}}) .$$

Ora vediamo il metodo per incollare i suddetti *hub scores* e *authority scores* relativi alle componenti individuali al fine di ottenere vettori di punteggio globali. Poichè la componente C di H contiene solo uno dei cinque *nodi hub*, il suo vettore stazionario dovrebbe essere pesato, ovvero moltiplicato, per $\frac{1}{5}$, mentre D, che contiene quattro dei cinque *nodi hub*, dovrebbe avere il suo vettore stazionario moltiplicato per $\frac{4}{5}$. In questo modo l'*hub vector* globale risulta:

$$\pi_h^T = \left(\overset{1}{\frac{4}{5}\cdot\frac{1}{3}} \quad \overset{2}{\frac{1}{5}\cdot 1} \quad \overset{3}{\frac{4}{5}\cdot\frac{1}{6}} \quad \overset{6}{\frac{4}{5}\cdot\frac{1}{3}} \quad \overset{10}{\frac{4}{5}\cdot\frac{1}{6}}\right) =$$
$$= (\overset{1}{0.2667} \quad \overset{2}{0.2} \quad \overset{3}{0.1333} \quad \overset{6}{0.2667} \quad \overset{10}{0.1333}) .$$

Con pesi simili per i *nodi authority* otteniamo il seguente vettore *authority* globale:

$$\pi_a^T = (\overset{1}{\frac{1}{4}\cdot 1} \quad \overset{3}{\frac{3}{4}\cdot\frac{1}{3}} \quad \overset{5}{\frac{3}{4}\cdot\frac{1}{6}} \quad \overset{6}{\frac{3}{4}\cdot\frac{1}{2}}) = (\overset{1}{0.25} \quad \overset{3}{0.25} \quad \overset{5}{0.125} \quad \overset{6}{0.375}) \, .$$

Confrontando i vettori *hub* ed *authority* ottenuti tramite SALSA con quelli ottenuti tramite HITS (pagina 49) vediamo che essi sono abbastanza diversi. Notiamo che la presenza di componenti connesse multiple (che si hanno quando G non è connesso) è una cosa molto positiva dal punto di vista computazionale perché le catene di Markov che devono essere risolte sono molto più piccole. Questo può essere in contrasto con la correzione che si opera nell'algoritmo Pagerank per un grafo web non connesso, dove l'irriducibilità è forzata aggiungendo connessioni dirette fra i nodi. Infine notiamo che altri pesi possono essere applicati per creare i punteggi globali.

Parte II

Capitolo 4

Quadro Generale

Come giá preannunciato, questa seconda parte tratterá di come presentare ad una classe delle scuole secondarie superiori uno degli algoritmi per l'*information retrieval* visti nei capitoli precedenti: il Google's Pagerank.
Ho lavorato con due classi di quarta superiore: la classe 4G del Liceo Scientifico Marconi di Pesaro e la classe 4A del Liceo Scientifico Archimede di San Giovanni in Persiceto. I professori con cui ho collaborato sono stati la Professoressa Vilma Polisca ed il Professor Stefano Accorsi, entrambi docenti di Matematica delle due classi, rispettivamente.
Quello che ho cercato di fare è spiegare a queste classi il funzionamento del motore di ricerca più usato al mondo e quindi far loro vedere un'applicazione della matematica che viene trattata a scuola. Innanzi tutto è bene avere un quadro generale ed esaustivo della situazione che abbiamo di fronte, ovvero è bene aver chiari:

1) qual è il il problema,

2) come rispondere a questo problema,

3) quali sono gli obiettivi (e non le finalitá) che ci prefiggiamo,

4) come valutare in itinere se questi obiettivi sono stati raggiunti,

5) come valutare alla fine del corso se gli obiettivi sono stati raggiunti.

Siccome ho lavorato con ragazzi che non avevano mai programmato, per me era anche importante verificare, passo per passo, se gli obiettivi erano o meno alla portata dei ragazzi, ovvero se stavo pretendendo troppo dai ragazzi dato il tempo che avevamo a disposizione.
Iniziamo ora la trattazione della sperimentazione rispondendo esattamente ai cinque punti enunciati sopra, in seguito, quando avremo un completo quadro generale, vedremo come i ragazzi hanno affrontato la sperimentazione (Capitolo 5) e trarremo le conclusioni sull'intero progetto (Capitolo 6), quindi non solo vedremo se i ragazzi hanno o meno raggiunto gli obiettivi ma anche come si potrebbe migliorare il progetto per poterlo riproporre in futuro.

4.1 Tematica del progetto

Quando ci si appresta a svolgere una sperimentazione (di qualsiasi natura, non solo scientifica) è importante aver chiaro lo scopo ed il motivo per cui si è scelto di intraprendere quel particolare percorso e non un altro, quindi la prima questione che vorrei analizzare riguarda l'intento del progetto stesso. Nella pratica quello che ho fatto è stato pormi inizialmente queste due domande:

- Quale è il problema?
- Perché ci interessa questo problema?

Rispondiamo alla prima domanda. La questione consiste in come mostrare ai ragazzi uno squarcio della matematica presente in uno strumento che utilizzano quotidianamente, matematica che coinvolge contenuti abitualmente non presenti nei percorsi scolastici come autovalori, autovettori, matrici stocastiche e matrici irriducibili. Google è un motore di ricerca che la maggior parte dei ragazzi crede di conoscere bene ma in realtà molti di loro non si sono mai chiesti cosa ci sia dietro la sua grande efficienza e velocità. Con questo progetto gli studenti potranno fare un uso più consapevole di questo importante strumento dato che studieranno le idee che stanno alla sua base.

Per mostrare ai ragazzi come queste idee sono state messe in pratica, è stato necessario introdurre dei concetti totalmente nuovi per loro che solitamente si trovano in un percorso universitario. È stato fondamentale valutare quanto approfonditamente questi concetti possono essere introdotti per essere facilmente compresi dai ragazzi senza rinunciare ad un'analisi matematicamente corretta dell'argomento. In definitiva, quello che ho fatto è stato ideare e organizzare un progetto didattico (suddiviso in unitá didattiche) per spiegare ai ragazzi delle due classi di quarta superiore il funzionamento di Google, ovvero per analizzare l'algoritmo che sta alla base del motore di ricerca. Nel capitolo 2 abbiamo incontrato quattro varianti di questo algoritmo e quello che a noi interessa, ovvero quello che verrà costruito insieme ai ragazzi, è il terzo, che porta alla risoluzione di un sistema lineare. Nel corso delle lezioni ho anche discusso con i ragazzi per riuscire ad individuare quali importanti differenze e innovazioni ci siano nei più recenti algoritmi sviluppati da Google, in particolare abbiamo parlato degli algoritmi Panda e Penguin usciti rispettivamente nell'agosto 2011 e nell'aprile 2012.
Focalizziamoci ora sulla seconda domanda. Ho trovato interessante questo problema per diversi motivi: innanzitutto si lavora con le matrici, ovvero i ragazzi hanno modo di vedere un utilizzo pratico e attuale delle matrici. Possono sperimentare la loro utilitá al di fuori del mero contesto scolastico. Inoltre si gettano le prime basi di programmazione ed è possibile introdurre in modo informale delle nozioni matematiche a livello universitario, osservando le reazioni dei ragazzi.
In parole povere quello che ho cercato di mostrare ai ragazzi è un'applicazione delle nozioni matematiche che loro conoscono; un'applicazione che li riguarda da vicino perché la maggior parte di loro, se non tutti, utilizza Google come motore di ricerca. Ho analizzato come reagiscono a lavorare con le matrici al di fuori del classico esercizio scolastico e come l'utilizzo del computer, in particolare di un software matematico, li possa aiutare in questo.
Più in generale il mio intento è quello di vedere sia quanto la tecnologia può aiutarli ad apprendere la matematica sia quanto può stimolarli. Lo stimolo

consiste nel presentare loro le basi della programmazione e del calcolo numerico, ovvero ho studiato come poter introdurre in una classe delle superiori le basi del calcolo numerico ed ho analizzato le reazioni degli studenti.

4.2 Sviluppo del progetto

Adesso che abbiamo chiarito la tematica della sperimentazione ed il perché di questa particolare scelta è ben soffermarci su come l'argomento sia stato trattato. Quello che ho fatto è stato dividere il progetto didattico in quattro unitá didattiche, ciascuna da due ore circa, per un totale di otto ore di lezione con ciascuna classe. Ho fatto sia lezioni frontali in classe, quindi classiche lezioni con l'ausilio della lavagna ed anche della LIM, che lezioni nel laboratorio di informatica. Durante le lezioni in classe ho spiegato in cosa consiste il Pagerank ed i problemi che si sono riscontrati per calcolarlo effettivamente, mentre in laboratorio, per poter costruire assieme ai ragazzi il codice dell'algoritmo e per poter spiegare loro il suo funzionamento, ho utilizzato un famoso software per l'analisi numerica: Matlab. Siccome nessuna della due classi aveva mai utilizzato Matlab per prima cosa ho mostrato loro i comandi base ed in seguito abbiamo sviluppato l'algoritmo. Durante i laboratori i ragazzi hanno lavorato in prima persona con Matlab, sia individualmente che a gruppetti. Quello che ho pensato di fare è dividere le otto ore che avevo a disposizione in due parti: prima una parte teorica, che occupasse le prime quattro ore, e poi una parte pratica.
La parte teorica si può dividere a sua volta in due sottoparti: la prima consiste in una introduzione al motore di ricerca Google, quindi la sua nascita e l'analisi dei risultati di una qualsiasi ricerca che facciamo; invece nella seconda parte si affronta il concetto di Pagerank, la sua formulazione matematica e la risoluzione dei due problemi principali legati all'esistenza e all'unicitá dell'autovalore $\lambda = 1$.
Per quanto riguarda la parte pratica, essa prende questo nome perché i ragazzi hanno utilizzato Matlab in prima persona. Anche questa parte si può

dividere in due sottoparti: durante la prima i ragazzi hanno appreso i principali comandi di Matlab che sono serviti loro per poter costruire, nella seconda sottoparte, l'algoritmo Google's Pagerank (chiaramente con il mio aiuto). Quindi possiamo dire che la prima sottoparte è propedeutica alla seconda.
Per non perdere troppo tempo nel scrivere alla lavagna i vari esempi di grafi e matrici o i comandi Matlab ho scritto due Power Point, che possiamo chiamare per semplicitá PP1 e PP2. In PP1 ho sintetizzato il concetto di Pagerank, quindi la sua definizione ricorsiva e come problema agli autovalori, inoltre ho trattato la costruzione dell'algoritmo Google's Pagerank ed ho aggiunto una sezione chiamata *Aggiornamenti*, che si può trovare nel Capitolo 2, in cui ho parlato dei principali miglioramenti effettuati sui due più recenti algoritmi di Google, Panda e Penguin. Invece PP2 è un Power Point strettamente relativo all'utilizzo di Matlab, quindi in esso si trovano i principali comandi che servono ai ragazzi per programmare in Matlab.
Ora siamo pronti per entrare nel dettaglio di ogni singola lezione. Lo schema sottostante presenta un sintesi di tutte e quattro le lezioni. Questo schema è quello che ho ideato inizialmente ovvero prima di iniziare la sperimentazione ed, in itinere, ho fatto qualche piccola modifica che illustreró nel capitolo relativo ai commenti dei ragazzi.

Struttura delle lezioni

Lezione 1**:**
Tempo: 2 ore.
Luogo: Laboratorio di informatica.
Strumenti utilizzati: Computers (uno per ragazzo) e collegamento Internet.
Contenuti trattati:

- nascita e storia del motore di ricerca,
- qual è il risultato di una (qualsiasi) ricerca, ovvero Google ci fornisce una lista di *links* secondo un particolare ordine,
- analisi dei risultati ottenuti (due analisi separate: prima si considera la lista ottenuta poi ci si sofferma sull'ordine dei *links*),

- introduzione del concetto di Pagerank, definizione ricorsiva.

Lezione 2**:**
Tempo: 2 ore.
Luogo: In classe.
Strumenti utilizzati: Lavagna e/o LIM e proiezione di PP1.
Contenuti trattati:

- definizione di Pagerank come problema agli autovalori,
- risoluzione problema 1: esistenza dell'autovalore $\lambda = 1$,
- risoluzione problema 2: unicitá dell'autovalore $\lambda = 1$,
- esercizi sulla costruzione di matrici di adiacenza e sui procedimenti per renderle (eventualmente) stocastiche ed irriducibili.

Lezione 3**:**
Tempo: 2 ore.
Luogo: Laboratorio di informatica.
Strumenti utilizzati: Computers (uno per ragazzo) e proiezione di PP2.
Contenuti trattati::

- panoramica sull'ambiente Matlab,
- comandi per la costruzione e gestione di matrici, vettori e scalari,
- cicli for, while e comando if,
- esercizi sulle matrici.

Lezione 4**:**
Tempo: 2 ore.
Luogo: Laboratorio di informatica.
Strumenti utilizzati: Computers (uno per ragazzo), collegamento Internet e proiezione di PP1.
Contenuti trattati:

- esercizio (costruzione di un semplice algoritmo),
- spiegazione dell'algoritmo Google's Pagerank che utilizza il metodo delle potenze,
- costruzione dell'algoritmo Google's Pagerank che risolve un sistema lineare,
- esercizio per determinare i ranghi delle pagine di un piccolo insieme web (al fine di utilizzare l'algoritmo appena creato).

4.3 Obiettivi

Fino ad ora abbiamo trattato di come impostare le varie unitá didattiche, di come fornire agli studenti una spiegazione più chiara e semplice possibile e delle finalitá di questo progetto.
Ora iniziamo a parlare degli *obiettivi*. In questa sede, mi sto attenendo alla definizione pedagogica di obiettivi ovvero intendo quei traguardi che ci aspettiamo e vogliamo siano raggiunti dai ragazzi durante e alla fine di un progetto. Per stabilire e descrivere gli obiettivi di un itinerario didattico bisogna individuare il contenuto dell'obiettivo ed esplicitare il tipo di prestazione, ovvero la competenza cognitiva, che si vuole stimolare su quel determinato contenuto. Per cui gli obiettivi sono molto diversi dalle finalitá: le finalitá hanno un'accezione più generale e spesso il professore è il soggetto; esse indicano cosa il professore deve riuscire a comunicare alla classe. Invece i soggetti, i protagonisti negli obiettivi sono gli alunni.
I traguardi che volevo che i ragazzi raggiungessero alla fine del progetto sono i seguenti:

1) definire il concetto di Pagerank,
2) costruire, dato un grafo, la corrispondente matrice di adiacenza (e viceversa),
3) riconoscere, partendo dai grafi, matrici stocastiche e matrici irriducibili,

4) applicare procedimenti per ottenere matrici stocastiche ed irriducibili,

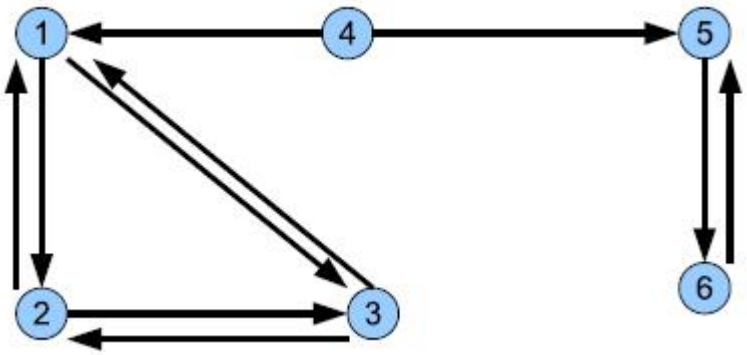
5) eseguire operazioni matriciali e piccoli algoritmi con Matlab,

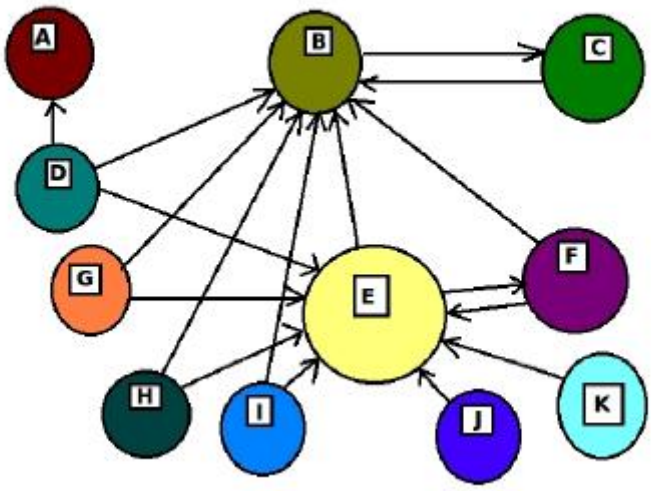
6) descrivere l'algoritmo Google's Pagerank,

7) descrivere le differenze tra questo algoritmo e quelli più recenti sviluppati da Google.

4.4 Valutazioni intermedie

Per poter meglio registrare ed analizzare le reazioni degli studenti e per controllare, lezione per lezione, se i ragazzi avevano raggiunto i primi obiettivi (quindi per accertarmi che non avessero troppi dubbi) ho deciso di realizzare due prove di valutazione intermedie, prima della prova finale. Come ho spiegato loro queste prove non sono pensate come dei compiti in classe ai quali viene messo un voto ma sono solo un modo, per me, di registrare cosa avessero capito o meno e cosí poter riprendere o approfondire dei concetti la lezione successiva. La prima prova di valutazione l'ho somministrata alla fine della seconda lezione. Essa è costituita dai due esercizi seguenti:





Per ciascun grafo costruisci la matrice di adiacenza associata.
Esse sono stocastiche? Sono irriducibili? Spiega il perché.
Rendile stocastiche se necessario.

In sostanza ai ragazzi venivano forniti due grafi (uno avente 6 nodi e l'altro 11) ed essi dovevano costruire le due matrici di adiacenza associate, dire se fossero stocastiche e/o irriducibili e modificarle in modo da renderle stocastiche se necessario. Non ho chiesto loro di renderle anche irriducibili (sempre se necessario) perché i calcoli sarebbero stati troppo pesanti. In questo modo ho potuto vedere se gli obiettivi 2) , 3) e 4) erano stati raggiunti.
Esercizi simili li ho proposti anche nella prova finale per controllare se ci fossero stati miglioramenti da parte di quegli studenti che avevano avuto problemi e anche per registrare eventuali peggioramenti perché, per esempio, alcuni studenti potevano giá aver dimenticato come passare dal grafo alla matrice di adiacenza.
La seconda prova intermedia è volta a valutare quanto i ragazzi abbiamo appreso dei comandi base di Matlab e ad analizzare come essi si rapportano di fronte alla creazione di un semplice algoritmo, quindi mi è servita per

vedere se i ragazzi avessero raggiunto l'obiettivo 5). Anche questa seconda prova è costituita da due esercizi. Il primo chiede di lavorare con le matrici; nel secondo, invece, gli studenti devono costruire un piccolo algoritmo che richiede l'utilizzo della condizione *if*. Qui sotto riporto il testo dei due esercizi:

Esercizio 1: *Considera la matrice* $A = \begin{pmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{pmatrix}$.
Una volta costruita elimina l'ultima riga e aggiungile la colonna a=[1;1], chiama B la matrice cosí ottenuta. Moltiplica B per la sua trasposta.
Chiama C la matrice cosí ottenuta. Calcola il massimo ed il minimo di ciascuna colonna di C e la somma dei suoi elementi per righe.

Esercizio 2: *Crea uno script per verificare se un numero* $\in \mathbb{N}$ *inserito da tastiera sia pari o dispari. CONSIGLIO: utilizza la funzione* rem *di Matlab.*

La funzione *rem* è una built-in function di Matlab che prende come parametri due numeri x,y $\in \mathbb{R}$ e restituisce il resto della divisione di x per y. perciò per determinare se un generico naturale n sia pari o dispari basta guardare se il resto della divisione di n per 2 sia 0. In questo caso il numero sarà pari altrimenti sarà dispari. I ragazzi hanno svolto il primo esercizio alla fine della terza lezione ed il secondo all'inizio della quarta.

4.5 Valutazione finale

Ho sottoposto la prova di valutazione finale qualche giorno dopo la conclusione della sperimentazione. Essa consiste in un questionario composto da dodici domande. Le prime sei chiedono ai ragazzi le loro opinioni sul corso, in particolare cosa è piaciuto e cosa potrebbe essere migliorato; quattro domande sono a scelta multipla e due sono domande aperte. Le restanti sei sono domande vogliono testare le competenze acquisite dai ragazzi. Tramite il questionario ho potuto vedere se gli obiettivi 1), 6) e 7) fossero stati rag-

giunti ed ho potuto avere un'ulteriore verifica riguardo agli obiettivi 2), 3), e 4). Ogni studente ha risposto individualmente alle domande ed il tempo che i ragazzi hanno impiegato per svolgere l'intero questionario è stato di circa 20 minuti. Ecco il questionario da me ideato:

QUESTIONARIO

1. Trovi che l'argomento presentato sia interessante (indipendentemente da come è stato svolto il corso) ?
   - Sí.
   - Abbastanza.
   - No.

2. Ti è piaciuto come è stato svolto il corso?
   - Sí.
   - Abbastanza.
   - No.

3. Qual è stata, tra le quattro lezioni svolte, quella che ti è rimasta più impressa? Perché?

   _____

   _____

   _____

   _____

4. Qual è stata, tra le quattro lezioni svolte, quella che ti è rimasta MENO impressa?Pperché?

   _____

   _____

   _____

   _____

5. Ti è piaciuto utilizzare Matlab?

   - Sí.
   - Abbastanza.
   - No.

6. Secondo te, il tempo dedicato a Matlab è stato:

   - troppo.
   - giusto.
   - troppo poco.

7. In base a quello che hai capito come definiresti il Pagerank?
   (Puoi utilizzare le formule viste o darne una definizione a parole).

   _____

   _____

   _____

   _____

8. Osserva le colonne della seguente matrice di adiacenza:

$$A = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

   Quale colonna è problematica? In cosa consiste il problema dal punto di vista del Pagerank e come si risolve?

   _____

   _____

   _____

9. Data la matrice di adiacenza $B = \begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

   ricostruisci il relativo grafo.

10. Se un tuo amico ti chiedesse come ottimizzare il ranking del proprio sito che consigli gli daresti?

11. Spiega che cosa è Matlab.

12. Questo è l'algoritmo *Pagerank.m* che abbiamo visto:

```
G = G - diag(diag(G));
[n,n] = size(G);
c = sum(G,1);
r = sum(G,2);
k = find(c~=0);
D = sparse(k,k,1./c(k),n,n);
e = ones(n,1);
I = speye(n,n);
x = (I - p*G*D)\e;
x = x/sum(x);
```

Spiega come funziona il codice (dai una spiegazione generale, non è necessario spiegare il codice riga per riga).

Come si può vedere, nelle domande aperte ho fissato il numero massimo di righe per la risposta in modo che i ragazzi avessero una maggiore guida.
Nelle domande in cui era richiesta la spiegazione di fatti matematici (come la 7, la 8 e la 12) ho esplicitamente detto ai ragazzi che, chi voleva, poteva utilizzare le formule viste insieme ma ciò non era necessario, bastava anche una spiegazione a parole. Dato che la sperimentazione tocca argomenti inusuali ed abbastanza complessi per dei ragazzi delle superiori, ho ritenuto che una spiegazione a parole fosse sufficiente per capire se i concetti ed i punti chiave fossero stati appresi.

Capitolo 5

In classe

Questo capitolo descrive ciò che è avvenuto durante le otto ore di lezione che ho fatto in ciascuna delle due classi di quarta superiore. Non parleró solamente dei contenuti da me trattati, anzi mi soffermeró soprattutto sui commenti e sulle opinioni espresse dai ragazzi. Il capitolo è diviso in due sezioni: nella prima parleró dell'esperienza che ho avuto nella classe 4G del Liceo Scientifico Marconi, nella seconda tratteró dell'esperienza avuta con l'altra classe, ovvero la 4A del Liceo Scientifico Archimede. In ciascuna sezione riporteró i commenti che gli studenti hanno fatto lezione per lezione.

5.1 4G Liceo Marconi

5.1.1 Lezione 1

La prima lezione che ho svolto con questa classe è avvenuta nel laboratorio di informatica. La professoressa Polisca (l'insegnante di matematica dei ragazzi) aveva giá comunicato loro che avremmo parlato dei motori di ricerca e che avremmo lavorato con un software matematico ma niente di più. Una volta che tutti gli studenti si sono seduti ed hanno avuto accesso ai computers con le loro credenziali, appena ho detto loro che potevano aprire un browser di navigazione, la prima cosa che hanno fatto è stata digitare l'indirizzo *www.google.it*. Questo fatto è molto indicativo, perché nessuno aveva

detto loro di aprire una determinata pagina o di cercare qualcosa. Ciò mi ha fornito un ottimo spunto per iniziare a parlare di quanto Google sia diffuso oggi e della sua storia. Prima di iniziare ad addentrarmi nella lezione vera e propria mi sono presentata, ho esposto brevemente quello che avremmo fatto nelle otto ore di lezione a nostra disposizione ed ho chiesto loro se avessero mai programmato o se avessero mai lavorato con Matlab. Quindici ragazzi su diciassette mi hanno risposto di no (la classe in totale era composta appunto da diciassette studenti); i due ragazzi che mi hanno risposto di sí sono appassionati di informatica ed hanno imparato molte cose da autodidatti.
A questo punto abbiamo iniziato a parlare di Google. Nessuno di loro sapeva i nomi dei due creatori del motore di ricerca nè aveva idea di quando il motore di ricerca fosse nato. Sono rimasti molto sorpresi di sapere che è un'invenzione abbastanza recente (circa 15 anni fa). Dopo aver raccontato loro come è nato Google abbiamo iniziato ad usarlo, ovvero ho fatto digitare loro una richiesta da fare al motore di ricerca ed ho chiesto di commentare il risultato che ottenevamo. Mi sembra importante far notare che all'inizio i ragazzi erano molto timidi, nessuno voleva rispondere alla mie domande, mentre il mio obiettivo era proprio fare una lezione dialogata, ovvero instaurare un dibattito con i ragazzi in modo che potessero emergere le loro idee riguardanti il funzionamento di Google. Attribuisco questo iniziale atteggiamento una delle tante clausole del contratto didattico (*' In classe non devo parlare, alla prof. non interessano le mie opinioni '*), ma dopo circa mezz'ora hanno iniziato a sentirsi più liberi e quindi a parlare.
Per quanto riguarda il commento del risultato che Google ci fornisce, dopo che abbiamo inserito una *query*, tutti hanno detto che il motore di ricerca ci restituisce una lista di *links* e che l'ordine di questi *links* non è casuale. Tutti sostenevano che: *'il primo link è il più importante'*. Abbiamo analizzato separatamente le due cose. Prima abbiamo discusso sulla lista di *links* che Google ci fornisce: volevo che arrivassero da soli, ma ovviamente con il mio aiuto, alla conclusione che Google ha un archivio di pagine web perché non può scansionare in tempo reale tutte le pagine del World Wide Web ogni

volta che qualcuno invia una richiesta. Per arrivare a ciò inizialmente ho fatto notare loro quanto poco tempo Google avesse impiegato per la ricerca, ed ho chiesto loro cosa cambiava se inviavano le loro 17 richieste contemporaneamente; tutti hanno risposto che non cambiava niente, il tempo di ricerca non era minimamente aumentato. Infine ho chiesto loro di cercare quante richieste venivano inviate a Google al secondo; un ragazzo ha trovato che ogni giorno venivano fatte a Google 70 miliardi di richieste e quindi facendo $(((70 \times 10^9) : 24) : 60) : 60$ hanno trovato quante richieste le persone di tutto il pianeta inviano, in media, ogni secondo. A questo punto ho posto la domanda cruciale: come fa Google a rispondere in cosí breve tempo a cosí tante richieste inviategli ogni secondo tenendo conto dell'abnorme numero di pagine web esistenti? Scansiona le pagine in tempo reale? L'intera classe mi ha risposto in coro di no ed un ragazzo ha ulteriormente commentato: *'Deve avere un archivio'*. In seguito abbiamo parlato di quanto spesso è aggiornato l'archivio e di quante pagine web può contenere.
Dalle loro risposte e reazioni deduco che i ragazzi non si erano mai posti la domanda se Google scansionasse le pagine in tempo reale o no ed anche dopo che abbiamo visto i numeri (riguardanti il numero totale di pagine web e quante persone mandano una richiesta a Google al secondo) non ho visto facce sorprese. Secondo me questo vuol dire che non hanno bene coscienza di quanto questi numeri siano alti, non riescono a rendersi conto degli ordini di grandezza. Anche se un ragazzo ha pronunciato la parola archivio non credo che la classe abbiamo pienamente interiorizzato che questo è l'unico modo possibile per avere una tale efficienza e velocitá nella ricerca. In seguito abbiamo parlato dell'ordine in cui vengono riportati i *links* e tutti gli studenti sapevano che, per Google, il primo *link* è più importante dell'ultimo.
Abbiamo fatto un brainstorming sui possibili criteri di importanza che Google può utilizzare oggi e, secondo i ragazzi, il motore di ricerca tiene conto:

- di quanto una pagina web sia visitata (più una pagina è visitata più è in alto nella lista),
- della corrispondenza fra la richiesta ed il titolo della pagina,

- dalle parole chiave presenti nel testo.

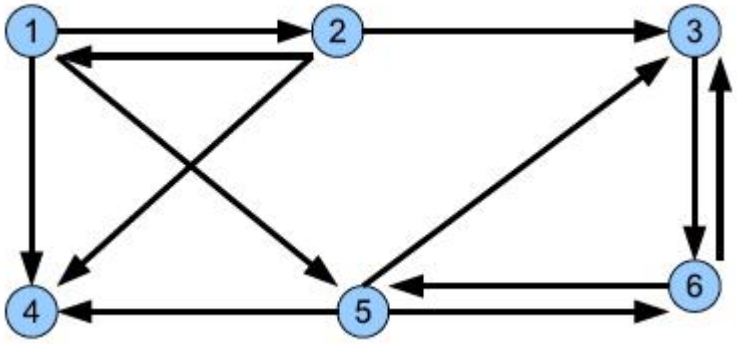
A nessuno è venuto in mente il Pagerank (come mi aspettavo) ma neanche l'intero contenuto del testo è emerso come criterio. Tutte queste idee sono molto valide e la prima e la terza sono effettivamente utilizzate da Google. Sono rimasta molto sorpresa quando i ragazzi hanno parlato delle parole chiave perché costituiscono un argomento molto attuale, infatti l'ultimo algoritmo di Google, Penguin, mira proprio ad eliminare il keywordstuffing.
A questo punto ho introdotto il Pagerank e, siccome dal brainstormig erano emerse ottime idee, ho subito introdotto i nuovi criteri che Google oggi utilizza nel suo algoritmo accanto al Pagerank. Su questa parte non volevo soffermarmi più tanto ma i ragazzi invece hanno espresso molto interesse su ciò e quindi abbiamo iniziato a trattare di Panda e Pengiun. Solo uno studente conosceva il nome dell'algoritmo che Google usa oggi e, con mia più grande sorpresa, solo questo studente sapeva il significato della parola algoritmo. Abbiamo parlato del keywordstuffing e del Pagerank (approssimativo) che fornisce la Google Toolbar. I ragazzi erano molto interessati a come migliorare, dal punto di vista del Pagerank, un sito web; mi hanno fatto molte domande a riguardo e ne discutevano vivacemente fra loro. Quando ho consigliato loro un sito che parlava di ciò lo hanno immediatamente visitato.
Dopo la ricreazione abbiamo visto il passaggio dalla definizione ricorsiva di Pagerank a quella come problema agli autovalori e lí il livello di partecipazione della classe è calato. Siccome mi è sembrato che la maggior parte della classe non avesse capito a fondo questo cambio di registro l'ho poi ripreso all'inizio della seconda lezione. Quando ho detto che ci mancava da verificare se la definizione fosse ben posta nessuno sapeva cosa questa espressione volesse dire. Prima di spiegare il significato ho chiesto cosa volesse dire secondo loro e mi aspettavo rispondessero: *'-non ci siamo ambiguitá, contraddizioni, ecc...'* invece il solo ragazzo che ha risposto ha detto: *'Non sia ricorsiva!'*. Allora gli chiesto perché, secondo lui, la ricorsione fosse un problema cosí grave e lui ha risposto in questo modo: *'Dato che la prima definizione che abbiamo dato era ricorsiva e abbiamo visto che era poco maneggevole, dob-*

biamo evitare che anche questa nuova definizione abbia lo stesso problema.' Anche se questa risposta è molto lontana dal significato dell'espressione 'definizione ben posta', l'ho trovata significativa perché dimostra lo sforzo di questo ragazzo nei confronti di un argomento per lui totalmente nuovo.
In seguito abbiamo costruito una matrice di adiacenza. Io ho scritto la prima colonna e ho chiesto loro di dettarmi la seconda; una ragazza mi ha risposto senza problemi. Abbiamo visto un anticipo della seconda lezione ovvero il problema legato all'esistenza dell'autovalore $\lambda = 1$ (una colonna di tutti zeri), ma, a mia sorpresa, per loro non era un problema, anzi una ragazza mi hanno detto: *'se ho una colonna di zeri vuol dire che quella pagina non ha outlinks e cosí almeno non perde parte del rango.'* Non riuscivano a capire come si potesse rimanere bloccati in una pagina che non ha *outlinks*, poi, discutendone insieme, ho capito a cosa stavano pensando: al tasto indietro. Ora era chiaro perché, per loro, arrivare in una pagina web senza *outlinks* non era un problema ma anzi era addirittura un vantaggio per il creatore della pagina: avendo il tasto indietro un utente non rimane bloccato in questa pagina ed inoltre essa avrá una alto rango perché lo non distribuisce ad altre pagine. A questo punto ho specificato che nel nostro web 'arcaico' il tasto indietro non esiste e quindi ci possiamo spostare solo seguendo i *links*.
Gli ultimi 5 minuti di lezione abbiamo di nuovo parlato del loro interesse principale, ovvero come migliorare il ranking di un loro ipotetico sito web.

5.1.2 Lezione 2

All'inizio della lezione abbiamo fatto un ripasso della lezione precedente durante il quale ho fatto vedere esplicitamente, cioè attraverso un esempio, l'equivalenza tra la definizione di Pagerank ricorsiva e la definizione come problema agli autovalori. Ho notato, con piacere, che i ragazzi conoscevano ed usavano senza problemi i simboli $\Rightarrow$ e $\Leftrightarrow$. Dalla loro reazione (ci sono stati diversi *'Oh è vero!'*) ho capito che, per la maggior parte di loro, l'equivalenza vista la scorsa lezione non era chiara; solo uno studente ha detto: *'Questo l'abbiamo giá fatto'*.

A questo punto abbiamo ripreso il primo problema legato alla definizione del Pagerank come problema agli autovalori: l'esistenza dell'autovalore $\lambda = 1$.
Il grafo che abbiamo preso in considerazione è il seguente:



Dopo aver rivisto dove stava il problema (il nodo 4 non ha *outlinks* e quindi la matrice di adiacenza ha una colonna di tutti zeri), ho chiesto come loro l'avrebbero risolto e le risposte sono state:

- *'eliminiamo la pagina 4'* ,
- *'sommiamo alla colonna 4 la 5 in modo da eliminare qualche zero, poi gli sommiamo la 2 e continuiamo finchè tutti gli zeri non sono stati sostituiti'* .

La prima idea è troppo frettolosa infatti, eliminando una colonna, si modificano le dimensioni della matrice e, di conseguenza, le dimensioni del web che stiamo analizzando e questa è una modifica troppo radicale. La seconda idea invece procede verso la giusta direzione, infatti ciò che si fa per eliminare la colonna di zeri è effettivamente una somma, ma una somma tra matrici, non tra vettori. Abbiamo visto che, nel caso specifico del nostro esempio, dobbiamo sommare la nostra matrice di adiacenza con una matrice delle stesse dimensioni fatta di tutti zero tranne gli elementi della colonna 4 che vanno presi della forma $\frac{1}{N}$ dove N è l'ordine della matrice.
Prima di iniziare a parlare del secondo problema alcuni ragazzi mi hanno rivelato che, qualche giorno prima del nostro secondo incontro, avevano creato un sito web e mi hanno chiesto se potevano riprendere il discorso su come

ottimizzare il ranking di un sito web cosí da poter migliorare il proprio sito. Questa loro iniziativa personale mi ha fatto enormemente piacere. L'idea è nata ad un gruppetto di 4 ragazzi della classe. Due si occupavano della programmazione, uno della parte economica (ovvero della pubblicitá, di come raccogliere fondi) ed il quarto aveva scritto il primo articolo per il sito (un articolo di carattere scientifico). Puntavano ad allargare il progetto a tutta la classe, cioè speravano che anche i loro compagni volessero scrivere articoli da pubblicare nel sito. Siccome questa seconda lezione l'ho svolta in classe, i ragazzi non mi hanno potuto mostrare subito il loro sito. Mi hanno dato maggiori dettagli sulla costruzione del sito nelle restanti due lezioni e abbiamo concordato che avremmo riparlato di come migliorare il ranking alla fine della quarta lezione, una volta costruito l'algoritmo.
Dopo questa parentesi abbiamo trattato il secondo problema, quello relativo all'unicitá dell'autovalore $\lambda = 1$. è stato più ostico capire come risolvere tale problema ma questa difficoltá ha permesso la nascita di un dibattito fra i ragazzi in cui ognuno diceva cosa aveva capito e cosa non era chiaro. Questo era proprio quello a cui miravo. Diversi ragazzi si chiedevano che ruolo avesse quell'α che appariva nella formula che rende la matrice stocastica P una matrice irriducibile. A questo punto ho parlato del legame che il Pagerank ha con la teoria delle probabilitá (ovviamente non ne ho parlato in modo approfondito e dettagliato) ed ho fatto un esempio di un'urna con quattro palline, due bianche e due nere, per evocare loro un elementare concetto di probabilitá. Tutti erano incuriositi dal perché i creatori di Google avessero scelto 0.85 come valore per α, molti di loro volevano dargli un valore diverso: un ragazzo lo voleva porre uguale ad 1 mentre un altro studente ha consigliato di prendere $\alpha = 0.90$. Come prima cosa io e la professoressa Polisca siamo intervenute per spiegare che prendere $\alpha = 1$ non è affatto utile. Ricordiamo che per ottenere una matrice irriducibile la matrice stocastica P deve essere modificata nel seguente modo:

$$A = \alpha P + (1 - \alpha)\frac{1}{n}ee^T \ ,$$

dove $(1-\alpha)\frac{1}{n}ee^T$ è una matrice che ha tutti gli elementi pari a $\frac{(1-\alpha)}{n}$.
Se prendiamo $\alpha = 1$ avremo che $1-\alpha$ sarà zero e quindi la seconda matrice sarà composta da tutti zeri per cui la nostra matrice stocastica P non verrebbe minimamente modificata ed il problema persisterebbe. Invece la seconda proposta è stata utile per far capire ai ragazzi che non bisogna assegnare ad α un valore troppo grande. Abbiamo visto che se passiamo da 0.90 a 0.99 la probabilitá di andare in un *link* alternativo, cioè costruito artificialmente da noi, sarà pari a 0.01, ovvero sarà una probabilitá troppo bassa che non ci permette di risolvere efficacemente il problema.
Un'altra difficoltá che i ragazzi hanno incontrato è stata capire il ruolo della matrice $(1-\alpha)\frac{1}{n}ee^T$ ovvero capire che cambiamenti venivano apportati al grafo costruendo la matrice $A = \alpha P + (1-\alpha)\frac{1}{n}ee^T$.
Prima di tutto ci siamo soffermati sulla definizione di grafo irriducibile. Ricordiamo che un grafo è irriducibile quando, dati due qualsiasi nodi (N_i, N_j), nel grafo esiste un percorso che porta da N_i a N_j. Quindi abbiamo visto che quello che dobbiamo fare è aggiungere artificialmente dei *links* che colleghino ciascuna pagina web ad un'altra. Il secondo passo è stato capire cosa significa moltiplicare lo scalare α per la matrice P ovvero come interpretare la matrice αP. Abbiamo visto che, siccome α è una probabilitá ed è abbastanza alta ($\alpha = 0.85$), moltiplicare α per P significa rendere i *links* originari più probabili di essere scelti da un qualsiasi utente, ovvero la probabilitá che un navigatore segua i *links* originari del grafo è più alta della probabilitá che segua uno dei nuovi *links* creati artificialmente da noi.
Infine abbiamo analizzato la matrice $(1-\alpha)\frac{1}{n}ee^T$. Grazie ad essa creo i nuovi *links* artificiali che mi permettono di collegare un qualsiasi nodo del grafo ad un altro e, come abbiamo giá detto, è meno probabile seguire uno di questi nuovi *links* rispetto ad uno dei *links* originari perché, in ciascun elemento della matrice, ritroviamo la probabiltá $1-\alpha$ che è una probabilitá bassa poichè vale 0.15. Prima di somministrare ai ragazzi la prima prova di valutazione intermedia ho fatto una breve introduzione su Matlab. Abbiamo parlato delle varie versioni di Matlab, della differenza tra le versioni

A e le versioni B ed ho spiegato loro perché noi avremmo utilizzato Matlab piuttosto che altri software scientifici. A questo punto ho consegnato loro la prova, abbiamo letto il testo degli esercizi insieme in modo che non ci fossero ambiguitá e che tutti i ragazzi avessero capito cosa gli esercizi chiedevano. Gli studenti hanno impiegato 20-25 minuti per fare entrambi gli esercizi.

5.1.3 Lezione 3

La terza lezione si è svolta nel laboratorio di informatica e, finalmente, i ragazzi hanno iniziato ad utilizzare Matlab. Invece di scrivere i vari comandi alla lavagna ho proiettato il Power Point PP2 in cui avevo giá inserito le nozioni principali che sarebbero servite agli studenti. La lezione è stata più dinamica (e più confusionaria) delle due precedenti in quanto i ragazzi non avevano mai lavorato con Matlab e solo 2 ragazzi su 17 avevano giá avuto esperienze di programmazione. Quando non sentivano o capivano un comando preferivano chiedere al compagno accanto piuttosto che a me e quindi c'era sempre una discreta agitazione. I ragazzi hanno lavorato individualmente con Matlab durante tutta la lezione ma, quando ho propostoloro il primo esercizio della seconda prova di valutazione intermedia, ho preferito farli lavorare a coppie.
All'inizio della lezione ho descritto l'ambiente Matlab: cosa significa tale nome, come è fatta la sua interfaccia grafica, che ruolo hanno le quattro finestre che la compongono e quale sia la differenza fra il linguaggio Matlab ed altri linguaggi (C, Python, ecc...). Dato che non avevano mai programmato ho brevemente trattato la differenza fra aritmetica reale e aritmetica finita, quest'ultima è quella in cui i calcolatori lavorano. Sono rimasti notevolmente sorpresi dal fatto che in Matlab, e quindi in aritmetica finita, non valga la proprietá commutativa. L'esempio che ho fornito loro è stato il seguente:

$$0.08 + 0.5 + 0.42 == 0.5 + 0.42 + 0.08 \ .$$

Il risultato di questa operazione logica (vi è ==) è 0, ciò vuol dire che l'uguaglianza non è verificata, non è vera in aritmetica finita.

In seguito abbiamo visto come creare delle variabili in Matlab: ho spiegato che i nomi scelti devono rispettare alcune regole di sintassi ed ho mostrato loro alcuni operatori di Matlab, in particolare ci siamo soffermati sugli operatori di base (addizione, sottrazione, ecc...) e su quelli relazionali (diverso, minore, minore uguale, ecc...). Abbiamo svolto insieme un esercizio di consolidamento molto semplice il cui testo chiedeva:

Introdurre 4 variabili e farne la media.

Abbiamo creato quattro variabili (a, b, c, d) e abbiamo dato loro i valori 10,20,30,40 rispettivamente. Fatto ciò abbiamo creato una nuova variabile, chiamata *media*, che contenesse la media dei quattro valori a, b, c, d ovvero abbiamo inserito il comando:

$$media = (a + b + c + d)/4$$

ed il risultato ottenuto è stato 25.
A questo punto abbiamo parlato di vettori e matrici. Ho mostrato come costruirli ed abbiamo usato alcune semplici built-in functions di Matlab che forniscono informazioni su vettori e matrici. Per esempio, dato un vettore v ed una matrice A, abbiamo visto la funzione $length(v)$ che indica la lunghezza del vettore v, la funzione $size(A)$ che fornisce la dimensione di A e la funzione *sum* che calcola la somma degli elementi di una matrice per colonne, quindi il risultato è un vettore. Inoltre abbiamo visto come estrapolare un singolo elemento ed un vettore (riga o colonna) da una matrice, abbiamo trattato la concatenazione di matrici, ovvero come aggiungere una riga o una colonna ad una matrice, ed infine come eliminare una qualsiasi riga o colonna. A questo punto ho presentato ai ragazzi la differenza tra *script* e *function*. Il primo script da noi creato consisteva nel copiare l'esempio visto precedentemente (la media fra quattro numeri) in un nuovo m.file chiamato *prova* e lanciarlo. Vorrei far notare che i due esercizi della seconda prova intermedia prevedono la realizzazione di due script e non di due function. Gli studenti, nell'intera sperimentazione, incontreranno solamente tre functions: *stoc*, creata da me per fornire ai ragazzi un esempio che riprendesse

la parte teorica e che permettesse loro di vedere come utilizzare il ciclo *for* e la condizione *if*; *pagerank*, l'algoritmo alla base del funzionamento di Google, e *surfer*, una funzione che, dati in input una pagina web e un numero naturale n, visita a partire da quella pagina n pagine web e costruisce la relativa matrice di connettivitá (ques'ultima funzione è scaricabile dal sito *http://www.mathworks.it/moler/*). Per permettere ai ragazzi di poter scrivere degli script abbastanza dinamici ed utili ho introdotto l'istruzione di ingresso *input* e l'istruzione di uscita *disp*. Prima di somministrare il primo esercizio della seconda prova di valutazione intermedia ho introdotto i cicli *for* e *while*. Questi due comandi, a differenza di tutti quelli precedentemente visti ed usati, hanno una grossa componente logica e teorica, essi sono alla base del concetto di iterazione. Non basta imparare a memoria la sintassi, bisogna capire come e quando è possibile usarli. Per questo motivo la loro spiegazione è stata la più problematica. Quando ho elaborato le slides su Matlab non ho pensato a questa importante componente concettuale legata a questi due comandi ed anche al comando *if* quindi immaginavo che la loro trattazione non sarebbe stata troppo lunga, invece ho dovuto soffermarmi molto sul ciclo *for* cosí, per non togliere tempo importante alla prova di valutazione, ho solamente introdotto il ciclo *while* non soffermandomi troppo. Un esempio che ho presentato ai ragazzi e che credo sia stato utile e chiarificatore per quanto riguarda il ciclo *for* è il seguente:

```
for i=1:m
    for j=1:n
        A(i,j)=i-j;
    end
end
```

Con questo codice viene creta una matrice avente m righe ed n colonne in cui il valore di ogni elemento è dato dalla differenza tra la riga i e la colonna j in cui l'elemento si trova. Per esempio con $m = 4$ e $n = 3$ otteniamo la

matrice:

$$A = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} .$$

Abbiamo visto che il primo ciclo *for* si occupa delle righe ed il secondo delle colonne e, per capire come è stata costruita la matrice, abbiamo seguito passo per passo quello che fanno i due cicli *for*. All'inizio, cioè al primo passo, abbiamo $i = 1$ ed $j = 1$ per cui l'elemento di posto $(1, 1)$ è zero. Al secondo passo abbiamo $i = 1$ e $j = 2$ quindi l'elemento di posto $(1, 2)$ sarà -1. Il secondo ciclo, quello in j, prosegue mentre il primo rimane fermo finchè $j = n$. A questo punto l'indice i viene incrementato ovvero $i = 2$ e si costruiscono gli elementi della seconda riga. Il procedimento continua in questo modo. A prima vista questo esempio potrebbe sembrare troppo difficile in quanto vi sono due cicli *for* annidati ma l'ho scelto perché i ragazzi avevano giá lavorato con la matrici e quindi, a mio parere, avrebbero fatto meno fatica a capire il concetto di iterazione lavorando con strumenti a loro giá familiari. Infine ho sottoposto loro il primo esercizio della seconda prova di valutazione intermedia.

5.1.4 Lezione 4

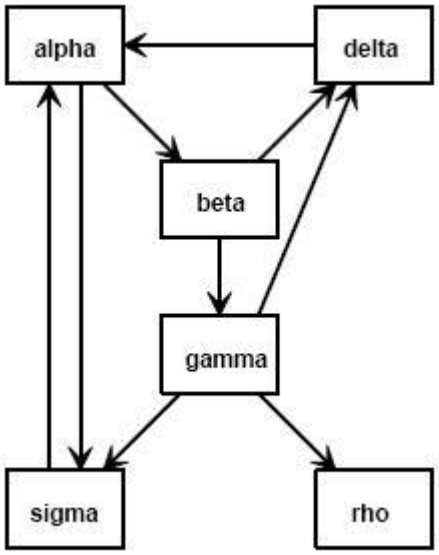
La quarta ed ultima lezione è stata svolta ancora una volta nel laboratorio di informatica. Siccome durante la lezione precedente non avevo fatto in tempo a spiegare il comando *if*, essenziale per lo svolgimento del secondo esercizio della prova di valutazione, ne ho parlato all'inizio della lezione. Mi è sembrato utile vedere l'applicazione del comando *if* all'interno di un codice che richiamasse qualche nozione e concetto visto nella parte teorica ed in particolare nella seconda lezione, cosí ho creato la seguente funzione *stoc*:

```
function [P]=stoc(A)
n=size(A);
n=n(1);
```

```
for j=1:n
    if A(:,j)==zeros(1,n)'
       P(:,j)=(1/n)*ones(1,n)';
    end
end
```

Questa funzione richiede una matrice quadrata come parametro di input ed ha come output sempre una matrice quadrata. Quello che fa è questo: se la matrice A passata come input ha una o più colonne in cui tutti gli elementi sono zero va a sostituire a questi vettori colonna dei vettori in cui ciascun elemento è della forma $\frac{1}{n}$ dove n è l'ordine della matrice. Quindi se come parametro le passiamo una matrice di adiacenza (ricordiamo che le matrici di adiacenza hanno tutti gli elementi non negativi) quello che otteniamo, ovvero il nostro output, sarà una matrice stocastica. In questa function, oltre alla condizione *if*, è anche presente un ciclo *for* che ci permette di controllare tutte le colonne della matrice. In conclusione mi è sembrato importante presentare questa funzione per tre motivi: ci permette di ripassare il ciclo *for*, ci fornisce un esempio di come utilizzare la condizione *if* ed infine fornisce un collegamento con la parte teorica precedentemente trattata.
A questo punto ho presentato ai ragazzi il secondo esercizio della seconda prova di valutazione. Mi sembrava che i ragazzi avessero tutti gli strumenti per poter svolgere l'esercizio al meglio ma, nei primi 5 minuti, solo una coppia di ragazzi aveva scritto qualche riga di codice mentre tutti gli altri guardavano lo schermo senza sapere che fare. ciò mi ha fatto capire che era necessario un ripasso dei comandi *input* e *disp* visti la lezione precedente ed inoltre quando ho chiesto ai ragazzi di andare a vedere sull'*help* di Matlab come utilizzare la funzione *rem* mi sono resa conto che non erano in grado di capire nè cosa facesse la funzione e nè come utilizzarla leggendo le istruzioni (scritte in inglese) fornite da Matlab. Allora ho spiegato loro perché servirci di *rem* e come utilizzarla. Non sono scesa nei minimi dettagli, ovvero non ho detto loro che come secondo parametro dovevano prendere il numero 2, perché altrimenti la difficoltá dell'esercizio mi sembrava azzerata.

Dopo lo svolgimento della prova siamo passati alla parte finale della sperimentazione: la creazione dell'algoritmo. Prima di tutto abbiamo visto come costruire la matrice di connettivitá che viene utilizzata negli algoritmi al posto della matrice di adiacenza. Non solo abbiamo trattato come poter scrivere queste matrici a mano ma anche come implementarle nel modo migliore in Matlab ovvero, come si può vedere nel Capitolo 2, sfruttando il comando *sparse* perché queste matrici, come del resto le matrici di adiacenza, sono sparse. L'esempio di web su cui ci siamo basati per costruire una matrice di connettivitá è il seguente:



La sua relativa matrice di connettivitá, chiamata G, è

```
G=
     0     0     0     1     0     1
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     1     1     0     0     0
     0     0     1     0     0     0
     1     0     1     0     0     0
```

Una volta scritto l'algoritmo Google's Pagerank lo abbiamo applicato a questo web per determinare i ranks delle 6 pagine che lo compongono.
A questo punto abbiamo iniziato ad addentrarci nell'ambito algoritmico. Prima di costruire l'algoritmo Google's Pagerank che determina i ranghi delle pagine risolvendo un sistema lineare, ho mostrato agli studenti l'algoritmo Google's Pagerank che sfrutta il metodo delle potenze. Ho fatto ciò per due motivi: primo perché volevo spiegare ai ragazzi cosa fosse il metodo delle potenze dato che ne avevamo parlato nelle prime due lezioni e, in secondo luogo, mi sembrava necessario fornire agli studenti una base su cui partire per poi andare a costruire l'algoritmo voluto. In sostanza i ragazzi hanno visto ben due algoritmi che permettono di determinare i ranghi di pagine web. In conclusione ho deciso di mostrarglieli entrambi perché il primo algoritmo (quello sfrutta il metodo delle potenze) è quello maggiormente legato alla parte teorica ovvero le modifiche che abbiamo apportato alla matrice di adiacenza le abbiamo fatte proprio per poter applicare il metodo delle potenze, mentre il secondo algoritmo è, a mio parere, di più facile comprensione per i ragazzi perché risolve un sistema lineare, argomento che avevano giá trattato, ed inoltre è più adatto per lavorare con matrici sparse. Siccome ora non utilizziamo più la matrice di adiacenza ma quella di connettivitá ho mostrato ai ragazzi il modo che Cleve Moler, l'ideatore di Matlab, usa per ottenere la matrice A, matrice stocastica ed irriducibile. Ricordiamoci che Cleve Moler scrive la matrice A in questo modo:

$$A = pGD + ez^T$$

dove D è la seguente matrice diagonale di elementi

$$d_{j,j} = \begin{cases} \dfrac{1}{c_j} & \text{se } c_j \neq 0 \\ 0 & \text{se } c_j = 0 \end{cases},$$

con c vettore riga le cui componenti sono la somma degli elementi di G per colonne, e è il vettore colonna di tutti 1, $p = 0.85$ e z è il vettore di

componenti

$$z_j = \begin{cases} \dfrac{1-p}{n} & \text{se } c_j \neq 0 \\ \dfrac{1}{n} & \text{se } c_j = 0 \end{cases} .$$

Per prima cosa abbiamo visto che la matrice stocastica ed irriducibile A ottenuta in questo modo è la stessa che ottenevamo partendo dalla matrice di adiacenza ovvero tramite la formula $A = \alpha P + \dfrac{(1-\alpha)}{n} ee^T$, ovviamente l'abbiamo visto solo in riferimento a questo particolare esempio. Successivamente ho mostrato agli studenti il primo dei due algoritmi per determinare il Pagerank, il quale sfrutta il metodo delle potenze. Il codice si trova nell'Appendice a pagina 87. Ho spiegato che le prime 9 righe di codice servono per creare la matrice A infatti viene definita la probabilitá p posta uguale a 0.85, vengono inizializzati i vettori c, e, z e la matrice D. Nella decima riga viene costruita la matrice A. Poi abbiamo parlato delle restanti righe del codice dove viene implementato il metodo delle potenze. Notiamo che questo è un metodo delle potenze semplificato perché sappiamo giá che l'autovalore dominante vale 1 quindi ci rimane solo da determinare il relativo autovettore. Dato che il ciclo *while* non l'avevo spiegato in modo approfondito non mi sono soffermata tanto sul metodo delle potenze. Ho cercato di trasmettere loro due aspetti del metodo che ritengo fondamentali: per prima cosa ho spiegato per quale motivo il metodo abbia questo nome e poi ho mostrato loro che, nel codice, non vengono salvati tutti gli iterati ma, ad ogni passo, si lavora solo con gli ultimi due. Questo meccanismo di sovrascrittura non è stato difficile da capire per i ragazzi. Quando ho chiesto perché, secondo loro, si procedesse in questo modo senza memorizzare tutti gli iterati uno studente mi ha risposto: *'Per risparmiare tempo.'* Secondo lui, memorizzando tutti gli iterati, l'algoritmo sarebbe stato più lento ovvero ci avrebbe messo più tempo a raggiungere la soluzione. Un altro ragazzo, che sapeva giá programmare, mi ha risposto: *'Per risparmiare memoria.'* Ovviamente la seconda risposta è quella giusta, ma ritengo che anche la prima fosse degna di nota. A questo punto eravamo pronti per costruire il secondo algoritmo Google's

Pagerank. Ho mostrato loro come ottenere il sistema lineare

$$(I - pGD)x = e$$

attraverso qualche piccola operazione algebrica, ho detto che per risolverlo in Matlab potevano usare il comando *backslash* ovvero scrivere

```
x = (I - p*G*D)\e
```

ed infine quello che ci mancava era creare tutte le variabili che entravano in gioco nel sistema lineare e, per fare ciò, bastava attenersi alle prime 9 righe dell'algoritmo precedente. Quello che ho fatto è stato scrivere il codice alla lavagna sotto dettatura dei ragazzi; io facevo domande sulle variabili da definire e loro mi rispondevano basandosi, appunto, sul codice giá visto. Il codice scritto si trova nell'Appendice a pagina 88. Non vi è stato tempo per trascrivere l'algoritmo in Matlab quindi ho fatto vedere ai ragazzi, utilizzando il mio portatile, cosa accadeva applicando tale algoritmo alla matrice G precedentemente costruita ovvero che Pageranks otteniamo per le 6 pagine web dell'esempio. Inoltre ho utilizzato la funzione *surfer* per costruire una nuova matrice di adiacenza di dimensioni maggiori a cui applicare l'algoritmo. Partendo dal sito *www.harvard.edu* ho chiesto che venissero mi visitate 50 pagine web raggiungibili da esso e cosí ho ottenuto una matrice di connettvitá di ordine 50 che rispecchiava una reale porzione di web. Infine, applicando a questa matrice l'algoritmo, ho mostrato ai ragazzi i ranghi delle 50 pagine web. Negli ultimi 10 minuti abbiamo ripreso la discussione relativa a come aumentare il ranking del loro sito. Ho mostrato ai ragazzi un interessante strumento creato da Google: Google Trends. Anche qui basta inserire una qualunque *query* ma il risultato che otteniamo è ben diverso: appare un grafico che mostra l'andamento della richiesta nel tempo, ovvero quanto, in questi ultimi anni, la richiesta sia stata 'googolata'. Scopriamo se tale richiesta è stata digitata più frequentemente da donne o uomini e se è stata cercata maggiormente da persone dai 15 ai 25 anni oppure tra i 25 e 35 anni e cosí via. Inoltre viene fornita anche la lista dei principali siti web che trattano quell'argomento. Chiaramente è possibile inserire come *query* un

indirizzo web e cosí si scoprono tante informazioni relative alla popolaritá di quel sito. Mentre la maggior parte degli studenti analizzava Google Trends con l'aiuto della Professoressa Polisca, ho chiesto ai quattro ragazzi che avevano creato un sito web di darmi maggiori informazioni sulla loro creazione. L'indirizzo del sito è *escapingbrains.altervista.org*; per costruirlo hanno utilizzato un server di hosting gratuito chiamato Altervista, il quale fornisce anche un *database* mysql che, ovviamente, è abbastanza limitato. Esiste anche la possibilitá di guadagnare qualche centesimo con le pubblicitá ed i ragazzi l'hanno inserita. Per gestire tutto il sistema dei contenuti utilizzano Wordpress, una piattaforma software di 'personal publishing' e content management system (CMS) che consente la creazione di un sito internet formato da contenuti testuali o multimediali, facilmente gestibili ed aggiornabili.
I quattro studenti si sono suddivisi i compiti in questo modo: due di loro amministrano Wordpress, perché sono gli unici che conoscono i linguaggi web come html, css, javascript e php, un ragazzo si occupa della pubblicitá ed il quarto aveva scritto il primo articolo. Una caratteristica importante che i ragazzi hanno voluto sottolineare è che questo progetto è libero e chiunque può partecipare, infatti il loro primo obiettivo era estenderlo alla classe e poi magari anche ad altri studenti della scuola interessati. Ancora non avevano inserito il sito sui principali e-media, come i social network, ma erano comunque raggiungibili da Google. Al suonare della campanella la lezione si è conclusa in modo divertente con un applauso di gruppo dato che si trattava della lezione conclusiva.

5.2 4A Liceo Archimede

Questa sezione è dedicata alla seconda sperimentazione che ho svolto nel Liceo Scientifico di San Giovanni in Persiceto nella classe 4A. Il docente di Matematica della classe è il professor Stefano Accorsi. A differenza della sperimentazione precedente non tutte le lezioni sono state svolte la mattina; la seconda e terza lezione le ho svolte di pomeriggio, dopo l'orario scolastico.

5.2.1 Lezione 1

La prima lezione si è svolta di mattina nel laboratorio di informatica ed era presente l'intera classe. In generale la lezione è stata molto simile a quella che ho svolto a Pesaro ma questi ragazzi erano molto meno timidi degli studenti della classe di Pesaro in quanto erano abituati a lezioni dialogate e a scambi di opinione. Abbiamo iniziato parlando dell'origine di Google e dei suoi creatori, poi abbiamo utilizzato Google (la richiesta che abbiamo inserito è stata il nome del loro professore) ed abbiamo discusso i risultati della ricerca. Abbiamo analizzato separatamente la lista di *links* ottenuta ed il loro ordinamento. Al contrario della precedente classe questi ragazzi erano consapevoli che, se tutti loro inviavano contemporaneamente una richiesta a Google, il tempo di risposta non cambiava minimamente. Un ragazzo ha anche esclamato: *'Se ogni volta che una persona fa una richiesta Google rallentasse sarebbe inutilizzabile!'* Egli voleva dire che se il tempo di risposta si allunga ogni volta che una persona si aggiunge nel fare una richiesta si dovrebbe aspettare cosí tanto tempo per avere una risposta che Google non avrebbe alcuna utilitá. Personalmente trovo che questa osservazione sia molto pertinente. In seguito ho chiesto agli studenti di cercare quante richieste vengano inviate a Google ogni secondo; dopo aver determinato la risposta i ragazzi avevano tutte le informazioni per capire se, fatta una richiesta, Google scansioni le pagine web in tempo reale o abbia un archivio.
Prima di rispondere gli studenti si sono interrogati su cosa volesse dire esattamente 'scansionare una pagina'; c'è stato un piccolo dibattito in cui vari studenti hanno detto cosa l'espressione volesse dire secondo loro. A questo punto non hanno avuto problemi nel rispondermi che Google deve avere un archivio. Questi studenti avevano maggior coscienza degli ordini di grandezza rispetto agli studenti della classe di Pesaro e credo che ciò abbia permesso ai ragazzi di capire a fondo la necessitá di avere un archivio ovvero che l'unico modo, per Google, per essere cosí efficiente e veloce è avere un archivio di pagine web. Inoltre abbiamo parlato della dimensione di tale archivio e di quanto spesso è aggiornato. A questo punto abbiamo analizzato il secondo

risultato: l'ordine dei *links* della lista. Prima di tutto abbiamo parlato di importanza e tutti gli studenti mi hanno detto che il primo *link* era più importante del secondo, terzo e cosí via. Poi ho chiesto quali, secondo loro, sono i possibili criteri di importanza che Google utilizza oggi. Il brainstorming ha portato ai seguenti risultati:

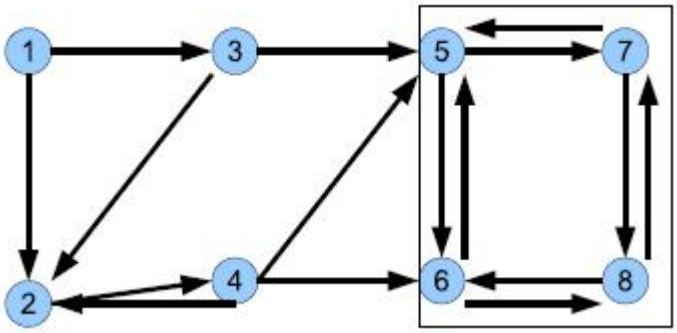
- il contenuto di una pagina web,
- le parole chiave presenti nel testo,
- la notorietá di una pagina web.

Sono stata molto contenta che il termine 'parole chiave' sia emerso una seconda volta. Inoltre ho chiesto maggiori informazioni sull'uso del termine 'notorietá' (di una pagina web). è avvenuto uno scambio di idee durante il quale una ragazza ha detto: *'più una pagina è visitata più è conosciuta.'* quindi abbiamo concordato che il numero di visite determina la 'notorietá' di una pagina. Dopo questo chiarimento posso dire che anche questo criterio era emerso nella classe precedente, invece il primo era una novitá.
A questo punto ho introdotto il concetto di Pagerank: ho spiegato che questo era il criterio originale che stava alla base del funzionamento di Google ed ho trattato la definizione ricorsiva. I ragazzi erano molto attivi e non esitavano a fare domande. Gli esempi sono stati fondamentali: alla fine della lezione un gruppo di ragazze mi ha detto che gli esempi erano stati chiarificatori ed avevano reso i concetti molto più semplici e comprensivi, inoltre, durante la lezione, ogni volta che proponevo un esempio diversi ragazzi alzavano la mano e ripetevano quello che io avevo detto con parole loro per poi chiedermi se avessero compreso correttamente. Negli ultimi 10 minuti abbiamo parlato dei nuovi algoritmi di Google, Panda e Pengiun, ed ho fatto loro notare che i primi due criteri di ordinamento che avevano proposto prima erano effettivamente usati in questi due algoritmi. Non abbiamo avuto tempo per trattare di come migliorare il ranking di un ipotetico sito web quindi ne abbiamo parlato durante la lezione successiva.

5.2.2 Lezione 2

Questa seconda lezione si doveva svolgere in classe ma alla fine abbiamo preferito spostarci in laboratorio dato che lí vi era la LIM. Questa lezione è avvenuta di pomeriggio e non era presente tutta la classe, purtroppo vi erano solo 14 studenti (l'intera classe è composta da 24 studenti). Come prima cosa abbiamo ripreso la definizione ricorsiva di Pagerank per poi affrontare la definizione come problema agli autovalori. A differenza della precedente sperimentazione, questa volta mi sono soffermata molto sul metodo delle potenze e sul perché dobbiamo aggiustare le definizione di Pagerank, ovvero dobbiamo essere certi che esista l'autovalore $\lambda = 1$ e che esso sia unico.
I ragazzi non hanno avuto difficoltá nel costruire la matrice di adiacenza; l'esempio su cui ci siamo basati è lo stesso che si trova nella Lezione 2 della precedente sezione. La matrice di adiacenza riferita a tale grafo l'ho costruita assieme ai ragazzi: io ho scritto alla lavagna la prima colonna della matrice e le altre cinque colonne sono state scritte da cinque studenti diversi. A questo punto siamo passati al primo problema: l'esistenza dell'autovalore $\lambda = 1$. Considerando ancora il grafo visto prima, ho fatto notare ai ragazzi che la pagina web 4 era differente dalle altre ovvero non aveva *outlinks* e, di conseguenza, la colonna 4 della matrice di adiacenza era costituita da tutti zeri. Ho chiesto agli studenti come mai questo era un problema specificando che l'unico modo che avevamo per muoverci nel web era tramite i *links* quindi non esisteva il tasto 'indietro' nè potevamo modificare l'indirizzo web. La risposta attesa è arrivata subito da una ragazza: *'Rimaniamo bloccati'.* Le due premesse sono fondamentali per la comprensione del problema infatti, nella sperimentazione precedente, vi erano state della incomprensioni proprio perché non avevo specificato che non esisteva il tasto 'indietro' nel web che stavamo considerando. Purtroppo quando ho chiesto ai ragazzi come avrebbero risolto il problema l'unica risposta che ho avuto è stata: *'Eliminiamo la pagina 4.'* che è una risposta inconcludente perché cosí facendo viene modificata la dimensione del web e quindi della matrice di adiacenza e ciò crea un ulteriore problema. Dopo aver mostrato la soluzione ed esserci soffermati

sulla stocasticitá della nuova matrice P abbiamo trattato il secondo problema: l'unicitá dell'autovalore $\lambda = 1$. Come nella sperimentazione precedente ho mostrato ai ragazzi il seguente grafo



ed ho detto loro che il problema risiedeva nella parte destra; a questo punto ho chiesto, secondo loro, quale fosse il problema. Dopo poco tempo un ragazzo mi ha risposto: *'Prima rimanevamo bloccati in una pagina ora invece in un gruppo di pagine.'* ed un altro studente ha aggiunto: *'Siamo bloccati in quella parte di grafo.'* Dopo aver parlato di grafi e matrici riducibili ed irriducibili ho mostrato loro come passare dalla matrice stocastica P alla matrice stocastica ed irriducibile A ovvero la formula: $A = \alpha P + \frac{(1-\alpha)}{n} ee^T$. Ovviamente anche questa volta sono stata bombardata da domande sul coefficiente α e sul ruolo della matrice $\frac{(1-\alpha)}{n} ee^T$ ed ho fornito le stesse spiegazioni date alla classe di Pesaro (Lezione 2, sezione precedente).
Quello che ho provato a fare in più rispetto alla sperimentazione precedente è stato dare una visione ancora più probabilistica del Pagerank introducendo il concetto di passeggiata aleatoria. Come sappiamo, navigare casualmente sul web significa fare una passeggiata aleatoria che, a sua volta, è un esempio di catena di Markov, quindi il rango di una pagina non è altro che la probabilitá di raggiungere quella pagina. Per spiegare la passeggiata aleatoria in modo divertente ho sfruttato il suo secondo nome: la passeggiata dell'ubriaco. Ho chiesto l'aiuto di uno studente volontario e gli ho domandato se poteva imitare la camminata di un ubriaco. Lui, molto giustamente, ha iniziato a spostarsi casualmente a destra e a sinistra e ciò mi ha permesso

di definire la passeggiata aleatoria (simmetrica) come una passeggiata in cui ho probabilitá $\frac{1}{2}$ di fare un passo a destra ed ancora probabilitá $\frac{1}{2}$ di fare un passo a sinistra. Ho anche accennato a passeggiate asimmetriche nelle quali si avrá maggiore probabilitá di andare a destra piuttosto che a sinistra o viceversa. Dopo questa spiegazione è stato facile per i ragazzi pensare al navigare sul web come ad una passeggiata aleatoria in cui fare un passo significa spostarsi da una pagina web ad un'altra e la probabilitá di spostarsi in una determinata pagina è data dal rango di questa. Trovo che questa presentazione ironica abbia avuto successo ed abbia permesso ai ragazzi di vedere il concetto in modo più semplice.
Prima della somministrazione della prima prova di valutazione intermedia abbiamo trattato di come migliorare il ranking di un ipotetico sito web ma l'argomento non ha ottenuto lo stesso successo avuto nella classe di Pesaro. Un ragazzo ha proposto di inserire tutti gli *outlinks* nella *home page*; quando gli ho chiesto perché l'avrebbe fatto ha risposto che cosí sarebbe stato più facile spostarsi nelle altre pagine del sito ed esterne al sito ma non aveva colto che cosí avrebbe disperso tutto il suo alto rango (l'*home page* di solito è la pagina con rango più alto in un sito web). Negli ultimi 20 minuti di lezione i ragazzi hanno svolto la prima prova di valutazione.

5.2.3 Lezione 3

La terza lezione si è svolta nel laboratorio di informatica ed i ragazzi hanno utilizzato individualmente Matlab. Anche questa lezione, come la precedente, è avvenuta di pomeriggio e non era presente tutta la classe; i ragazzi erano circa 15. Questa lezione è stata la più difficile per me perché, a differenza della precedente classe dove due studenti sapevano giá programmare, qui nessun ragazzo aveva mai programmato. Questa apparentemente piccola differenza è stata significativa perché nella classe di Pesaro i due ragazzi che sapevano giá programmare aiutavano i loro compagni che avevano più difficoltá, in una certo senso si può dire che 'trainassero' la classe, mentre ora non vi erano figure di questo tipo e ciò ha cambiato molto la situazione. Inoltre molti

studenti erano inevitabilmente stanchi dato che avevano alle spalle diverse ore di lezione. In generale per i ragazzi Matlab è risultato difficile.
La parte iniziale è stata analoga a quella svolta nella classe 4G di Pesaro ovvero ho introdotto l'ambiente Matlab, ho parlato della sua interfaccia grafica, di come definire le variabili ed infine della differenza tra aritmetica reale e aritmetica finita. Ho mostrato ancora una volta un esempio che ci permette di capire che in aritmetica finita non vale la proprietá commutativa e, come mi aspettavo, tale esempio li ha sorpresi molto. Molti ragazzi mi hanno chiesto come mai ciò accadesse ma, purtroppo, non abbiamo potuto soffermarci sull'argomento dato che non vi era tempo per trattare anche gli errori di arrotondamento e troncamento. Quando abbiamo iniziato a parlare di vettori e matrici la situazione si è complicata. Pur avendo trattato le matrici in classe i ragazzi non sapevano nè cosa fosse la trasposta di una matrice nè che per fare la moltiplicazione tra due matrici di ordine $m \times n$ e $a \times b$ è necessario che $n = a$. Non hanno avuto problemi con questi due nuovi concetti infatti, dopo aver scritto alla lavagna due matrici, A e B, di ordine 2×3 un ragazzo mi ha detto (prima che io potessi fare una qualsiasi domanda): *'$A \times B$ non lo posso fare ma $A^T \times B$ invece sí, giusto?'*
La cosa difficile per i ragazzi è stata lavorare con le matrici utilizzando Matlab. Hanno imparato presto a creare matrici ma la loro principale difficoltá è stata modificarle ovvero aggiungere e togliere una riga o una colonna ed estrarre un vettore riga o colonna. Consideriamo, per esempio, un vettore $v = (7, 8, 9)$ ed una matrice A fatta in questo modo:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 0 & 10 \\ 1 & 7 & 19 \end{pmatrix} .$$

Se voglio aggiungere il vettore v come ulteriore riga di A posso fare:

```
B=[A;v];
```

e cosí ottenere la matrice

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 0 & 10 \\ 1 & 7 & 19 \\ 7 & 8 & 9 \end{pmatrix}$$

che coincide con la matrice A ma alla quale ho aggiunto una riga. Per aggiungere ad A il vettore v come colonna posso fare:

```
C=[A,v'];      .
```

A diversi studenti creava confusione il fatto di mettere la virgola o il punto e virgola a seconda di quello che si voleva aggiungere (righe o colonne) ed il fatto di considerare v o v trasposto. Prima si somministrare loro il primo esercizio delle seconda prova di valutazione ho trattato il ciclo *for*. Ho utilizzato lo stesso esempio che avevo proposto alla classe precedente, ovvero come scrivere una matrice mediante il ciclo *for* (Lezione 3, sezione precedente). Ho spiegato loro che servono due cicli uno dentro l'altro perché una matrice è un array di dimensione 2 e quindi è necessario un ciclo per le righe ed uno per le colonne. Una cosa che ho fatto solo in questa classe e non in quella di Pesaro è stato far provare direttamente ai ragazzi a costruire, in Matlab, una nuova matrice scritta alla lavagna utilizzando il ciclo *for*. La matrice considerata era la seguente:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \end{pmatrix} \quad .$$

Gli studenti, lavorando a gruppetti di due o tre persone, hanno scritto correttamente il codice che è il seguente:

```
for i=1:3
    for j=1:3
        M(i,j)=i+j;
    end
end
```

L'unica modifica che andava fatta rispetto al codice fornito nell'esempio era la condizione nella terza riga. A questo punto ho sottoposto ai ragazzi il primo esercizio della seconda prova di valutazione. Siccome, fino al quel momento, avevano avuto difficoltá con Matlab ho lasciato loro circa mezz'ora per svolgere l'esercizio cosí che tutti avessero tempo per pensare e fare vari tentativi. Al termine della lezione alcuni studenti mi hanno detto di aver trovato difficile l'esercizio e la lezione in generale.

5.2.4 Lezione 4

L'ultima lezione l'ho svolta di mattina nel laboratorio di informatica. Dato che nelle lezioni pomeridiane c'era stato un calo di affluenza, appena entrata in classe mi sono resa conto che 4/5 ragazzi avevano seguito solo la prima lezione e 2/3 ragazzi non li avevo mai visti quindi, come prima cosa, ho deciso di fare un riassunto della parte teorica per quest'ultimi. Per fare il riassunto ho chiesto l'aiuto di due ragazze, Caterina e Carolina, che erano state presenti a tutte le lezioni e che ero sicura avessero capito bene cosa avevamo fatto. Come prima cosa ho ripreso la definizione di Pagerank, l'ho scritta come problema agli autovalori e Caterina ha ricordato agli altri compagni come costruire la matrice di adiacenza. Poi ho parlato dei due problemi giá incontrati ovvero l'esistenza e l'unicitá dell'autovalore $\lambda = 1$. Carolina ha ricordato alla classe come risolvere il primo problema e successivamente ho parlato delle matrici irriducibili soffermandomi sul fatto che, modificando le matrici, noi creiamo artificialmente nuovi *outlinks*. Abbiamo anche parlato di α come probabilitá di seguire i vecchi *links*, ovvero quelli originali del grafo, e di $1 - \alpha$ come probabilitá di seguire i nuovo *links* aggiunti da noi. Ho brevemente ripreso come costruire una matrice in Matlab ed infine ho trattato qualche comando nuovo di Matlab, nello specifico il ciclo *while* ed la condizione *if*. Al contrario della sperimentazione precedente ora ho dedicato più tempo al ciclo *while* mostrando ai ragazzi diversi esempi tramite il Power Point, da me creato, PP2. Dopo aver spiegato la condizione *if*, utilizzando ancora una volta la funzione *stoc* da me creata, ho proposto ai ragazzi il

secondo esercizio della seconda prova di valutazione intermedia.
Per rimediare agli sbagli della lezione scorsa, quando i ragazzi mi avevano detto che l'esercizio che avevo dato loro era molto difficile ed anche dopo l'esperienza avuta con la classe di Pesaro, li ho guidati molto nello svolgimento dell'esercizio. La prima riga del codice

$$n = input('inserisci\ un\ numero\ naturale:\ ');$$

l'ho scritta alla lavagna ed ho anche detto loro cosa fa la funzione *rem* e come usarla, ovvero ho spiegato che la funzione richiede due parametri in input, che come risultato fornisce il resto della divisione del primo numero per il secondo e che, nel nostro caso, come secondo parametro dobbiamo utilizzare 2 poichè un numero pari è, per sua natura, divisibile per 2 cioè il resto della divisione di quel numero per 2 è 0. Quest'ultimo fatto non l'avevo specificato agli studenti della precedente sperimentazione ma, in questo caso, ho creduto fosse meglio fornire questo ulteriore aiuto dato che i ragazzi avevano giá manifestato le loro difficoltá con Matlab. Infine ho ricordato loro come usare il comando *disp*. L'unica difficoltá dell'esercizio era scrivere correttamente la riga di codice dove era presente la condizione *if*. Dopo l'esercizio ho ripreso la definizione di matrice di connettivitá, soffermandomi sul fatto che tali matrici sono sparse ed ho subito mostrato loro un modo efficiente per memorizzare matrici sparse in Matlab, ovvero utilizzando il comando *sparse*. Il grafo su cui ci siamo basati per costruire una matrice di connettivitá si può trovare nella Lezione 4 della precedente sezione.
Dopo aver mostrato il modo che Cleve Moler usa per scrivere la matrice A stocastica ed irriducibile ovvero $A = pGD + ez^T$ servendosi della matrice di connettivitá G e non della matrice di adiacenza (i dettagli si possono trovare nella Lezione 4 della precedente sezione) abbiamo verificato, tramite un esempio, l'equivalenza di questa scrittura con quella vista da noi $A = \alpha P + \dfrac{(1-\alpha)}{n} ee^T$. A questo punto eravamo pronti per trattare il codice *Pagerank.m* che sfrutta il metodo delle potenze.
Ho deciso di dedicare più tempo al metodo delle potenze rispetto a quanto avevo fatto con la classe precedente. Ho spiegato perché il metodo avesse

questo nome quindi ho scritto alla lavagna la successione degli iterati dove compaiono le potenze di A. Ho potuto analizzare meglio la condizione del ciclo *while* dato che lo avevo trattato in modo più approfondito ed infine, come avevo fatto nella classe 4G del Liceo Scientifico di Pesaro, ho fatto notare ai ragazzi che nel codice non vengono salvati tutti gli iterati ma vengono memorizzati solo gli ultimi due quindi ad ogni passo, cioè ad ogni entrata nel ciclo while, si sovrascrivono i valori. Gli studenti di questa classe hanno avuto più difficoltà a comprendere questo meccanismo di sovrascrittura, infatti un ragazzo mi ha chiesto: *'Ma cosí facendo, cioè sovrascrivendo ogni volta, non si crea confusione?'* Ho risposto che, apparentemente, questo meccanismo poteva creare confusione nelle persone che, come loro, non aveva mai programmato ma che, dal punto di vista del codice, del suo funzionamento, non vi era alcuna confusione perché ogni volta si aggiornavano gli iterati e si memorizzavano solo i due più 'nuovi', ovvero solo quelli che servivano per il passo successivo. A questo punto ho chiesto per quale motivo, secondo loro, si preferisse agire in questo ed una delle due ragazze che mi aveva aiutato nel ripasso a inizio lezione mi ha risposto: *'Se invece di salvare tutti gli iterati teniamo sempre solo gli ultimi due andremo a memorizzarci molte meno cose.'* Questa risposta, chiaramente detta in un linguaggio da studente, è corretta, infatti sovrascrivendo gli iterati si risparmia molta memoria. Infine abbiamo costruito l'algoritmo con il sistema lineare in modo analogo a come l'avevo costruito con la classe 4G di Pesaro. Ho fatto ricopiare ai ragazzi il codice perché volevo che sperimentassero in prima persona l'algoritmo usando la matrice di connettività G che avevamo creato prima. Questa fase di trascrizione (l'algoritmo l'avevo scritto alla lavagna come nella sperimentazione precedente) è stata molto lunga. Dopo che tutti gli studenti erano riusciti a far girare correttamente il codice, ho mostrato loro la funzione *surfer* dandole sempre come parametri la pagina web *www.harvard.edu* ed il numero 50 in modo da ottenere una matrice di connettività di ordine 50.
Anche questa quarta lezione si è conclusa in modo divertente con un applauso di gruppo dato che si trattava della lezione conclusiva.

Prima di passare al prossimo capitolo, riassumiamo i concetti base di Algebra Lineare e Calcolo Numerico che i ragazzi hanno visto ed appreso in queste lezioni, concetti che sono serviti come strumento per la spiegazione del funzionamento del motore di ricerca Google e, in particolare, per l'analisi dell'algoritmo Google's Pagerank. Inoltre riassumiamo le differenze principali che vi sono state fra le due sperimentazioni.
La tabella seguente mostra le definizioni, i teoremi e i metodi introdotti al fine di condurre un'accurata analisi del concetto di Pagerank uniti ad alcune operazioni e comandi relativi al programma Matlab necessari per la comprensione dell'algoritmo.

CONCETTI INTRODOTTI

Algebra Lineare:

1) definizione di autovalore e autovettore
2) definizione di matrice sparsa
3) definizione di matrice stocastica
4) definizione di matrice irriducibile
5) definizione di grafo fortemente connesso
6) teorema di Perron-Frobenius

Calcolo Numerico:

1) ciclo for, while e istruzione if
2) metodo delle potenze

Matlab:

1) operatori di base e relazionali
2) costruzione di scalari, vettori e matrici
3) funzioni di matrici
4) concatenazioni di matrici
5) condizioni logiche e operazioni aritmetiche su vettori e matrici

Per quanto riguarda le differenze fra le due sperimentazioni, qui di seguito riporto le principali diversità che ho riscontrato sia dal punto di vista organizzativo (orari e conoscenze preliminari dei ragazzi) sia per quanto riguarda le reazioni e le risposte delle due classi al progetto, ovvero gli interessi che i

ragazzi hanno mostrato, le difficoltà che hanno avuto, ecc...

PRINCIPALI DIFFERENZE DELLE DUE SPERIMENTAZIONI:

1. orari diversi: tutte le lezioni svolte nella prima classe hanno avuto luogo la mattina, mentre nella seconda sperimentazione due lezioni (la seconda e la terza) sono state svolte nel pomeriggio e ciò ha causato una minore attenzione da parte dei ragazzi e una minore frequenza;

2. nella prima classe vi erano due studenti che sapevano programmare e che hanno dato un'importante contributo durante il lavoro con Matlab, al contrario nella seconda classe nessuno sapeva programmare;

3. i ragazzi della prima classe erano molto interessati ai risvolti pratici e alle caratteristiche degli odierni algoritmi di Google, quindi con loro ho approfondito maggiormente questa parte;

4. i ragazzi della seconda classe hanno lavorato meglio con le matrici ovvero hanno avuto meno difficoltà relativamente agli argomenti della seconda lezione;

5. i ragazzi della seconda classe hanno avuto più difficoltà iniziali con Matlab e, per questo motivo, con loro ho speso più tempo nell'analisi dell'algoritmo Google's Pagerank.

Capitolo 6

Conclusioni

In questo capitolo analizzeremo i risultati delle prove intermedie e della prova finale (questionario) che ho sottoposto agli studenti delle due classi nel corso della sperimentazione. Infine trarremo le conclusioni sull'intero progetto ovvero vedremo quali obiettivi sono stati raggiunti dai ragazzi e quali no e parleremo di eventuali modifiche da fare per migliorare l'intero progetto.

6.1 Analisi prima prova intermedia

Ho deciso di suddividere le varie prove consegnatemi dai ragazzi in cinque categorie in base al numero di esercizi corretti che hanno svolto. Ricordiamo che la prima prova intermedia è composta da due esercizi da svolgere 'a mano' entrambi riguardanti la costruzione di matrici. Il testo degli esercizi si trova nel Capitolo 3 a pagina 68. Detto ciò, le cinque categorie in cui ho suddiviso i compiti dei ragazzi sono: *Entrambi gli esercizi corretti, Solo il primo corretto, Solo il secondo corretto, Entrambi sbagliati, Non classificabili.* L'ultima categoria comprende quei compiti che sono stati lasciati in bianco o quasi. È importante notare una fondamentale differenza fra i compiti non classificabili della classe di Pesaro e quelli della classe di San Giovanni in Persiceto: i ragazzi della prima classe che hanno lasciato i compiti quasi in bianco sono stati costretti a farlo per una questione di tempo, in quanto

avevano un permesso speciale per uscire prima da scuola al fine di prendere l'autobus per tornare a casa, invece nella seconda classe tutti i ragazzi hanno avuto lo stesso tempo (20 minuti) per poter svolgere la prova quindi quegli studenti che non hanno svolto alcun esercizio l'hanno fatto volutamente. Di seguito riporto lo schema che rappresenta la classificazione da me fatta.

Prova intermedia 1	Classe 1	Classe 2
Totale compiti	15	11
Entrambi gli esercizi corretti	5	5
Solo il primo corretto	6	3
Solo il secondo corretto	0	0
Entrambi sbagliati	2	1
Non classificabili	2	2

Dalla tabella emerge che, mentre quasi tutti i ragazzi della classe di Pesaro erano presenti allo svolgimento di questa prima prova (15 su 17), solo 11 dei 24 ragazzi appartenenti alla classe di San Giovanni erano presenti. Vediamo che $\frac{1}{3}$ dei ragazzi della prima classe ha svolto entrambi gli esercizi correttamente, mentre nella seconda classe quasi la metà.

Ora analizziamo in modo più approfondito ciascuna categoria partendo da *Entrambi gli esercizi corretti*. Questi 10 ragazzi (5 della prima classe e 5 della seconda) non hanno avuto alcun problema nello svolgimento della prova. Entrambe le matrici di adiacenza che hanno costruito sono corrette ed hanno scritto che la prima di tali matrici è stocastica e riducibile mentre la seconda non è stocastica ed è riducibile. Quest'ultima matrice è stata correttamente modificata per essere resa stocastica, ovvero hanno precisato che la prima colonna va sostituita con un vettore 11×1 avente tutte le componenti pari a $\frac{1}{11}$. Per quanto riguarda le motivazioni date, ovvero giustificare perché tali matrici siano o no stocastiche ed irriducibili, le risposte sono state differenti. Uno dei dieci ragazzi (appartenente alla prima classe) non ha dato alcuna giustificazione di ciò che ha scritto. Gli altri nove hanno tutti detto per quale motivo le due matrici sono riducibili: 5 hanno semplicemente scritto che nei grafi erano presenti dei loop, mentre gli altri 4 hanno specificato esattamente

dove si trovano questi loop. Due esempi di risposte sono:
Ragazzo della classe di Pesaro: *'(La prima matrice) è riducibile perché ci sono dei loop, (la seconda) perché c'è un loop.'*
Ragazza della classe di San Giovanni: *'La (prima) matrice è riducibile poiché ci sono due loop: tra 1, 2, 3 e tra 5, 6. La (seconda) matrice è riducibile poiché c'è un loop tra B e C.'*
Tutte le altre risposte sono simili a queste due. Notiamo che tutti e quattro i ragazzi che hanno precisato dove fossero i loop appartengono alla classe di San Giovanni. Meno omogenee sono state le motivazioni riguardanti la stocasticità delle matrici. Due studenti su dieci non hanno scritto perché la prima matrice sia stocastica e la seconda no. Le altre otto risposte si possono dividere in due gruppi in base a quello che gli studenti hanno scritto in merito alla prima matrice; vi è il gruppo degli studenti che scrive *'La (prima) matrice è stocastica perché non possiede una colonna di tutti zero'* e il gruppo che ha risposto *'La (prima) matrice è stocastica perché la somma degli elementi di ciascuna colonna è 1'*. Il primo gruppo è composto da 6 studenti e la loro risposta non è corretta. Dalla definizione di matrice stocastica possiamo dire che se una matrice presenta una colonna di zeri allora non è stocastica ma una matrice può non essere stocastica pur non possedendo una colonna di zeri, infatti, per esempio, tale matrice

$$\begin{pmatrix} 1 & 0 & \frac{1}{2} \\ 1 & 1 & 0 \\ 1 & 0 & \frac{1}{2} \end{pmatrix}$$

non è stocastica perché la somma degli elementi della prima colonna è > 1. Tutti gli studenti che hanno dato tale risposta hanno poi motivato correttamente il fatto che la seconda matrice non sia stocastica perché, in questo caso, è corretto dire che se la matrice ha una colonna di zero allora non è stocastica. Solo due studentesse hanno ricordato la corretta definizione di matrice stocastica e quindi hanno fornito la seconda motivazione che è corretta. Tali studentesse hanno giustificato correttamente anche la loro seconda risposta.

Non ho considerato errore il fatto che gli studenti non abbiamo precisato che una matrice per essere stocastica deve anche avere elementi non negativi perché noi abbiamo sempre lavorato con matrici ≥ 0.
Ora passiamo alla seconda categoria che corrisponde a quella dei ragazzi che hanno svolto correttamente solo il primo esercizio. È bene notare che quasi tutti questi studenti (8 su 9) hanno provato a scrivere la seconda matrice di adiacenza ed hanno correttamente detto che tale matrice non è né stocastica né irriducibile. Questa categoria si può suddividere in tre gruppi: chi lascia l'esercizio incompleto, chi sbaglia la matrice di adiacenza, chi sbaglia la matrice e lascia l'esercizio incompleto. Tre studenti su nove appartengono al primo gruppo. Questi ragazzi pur scrivendo (correttamente) che la seconda matrice non è stocastica ma non hanno apportato alcuna modifica al fine di renderla tale. Il secondo gruppo è composto da due studenti che hanno scritto male la matrice di adiacenza. Molto probabilmente hanno sbagliato a contare il numero di *outlinks* che uno degli nodi del grafo aveva, ma hanno scritto correttamente la modifica da apportare alla matrice per renderla stocastica. Per esempio, uno dei due studenti, scrive la quinta colonna della matrice in questo modo:

$$(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)\,,$$

mentre il vettore corretto è:

$$(0 \quad \frac{1}{2} \quad 0 \quad 0 \quad 0 \quad \frac{1}{2} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)\,,$$

quindi non ha notato che il nodo E aveva due *outlinks*. Questo errore non lo ritengo grave in quanto si tratta di un errore di distrazione: i ragazzi non hanno osservato il grafo attentamente e quindi è sfuggito loro qualche collegamento. Infine abbiamo il terzo gruppo che è quello più numeroso, formato da quattro dei nove studenti. Essi hanno sia sbagliato la matrice di adiacenza, non prestando attenzione al grafo, sia lasciato l'esercizio incompleto, non precisando quale modifica andasse fatta per rendere tale matrice stocastica. Una ragazza appartenente a tale gruppo non ha nemmeno provato a scrivere

la matrice di adiacenza. Credo che sia importante precisare che sei di questi nove ragazzi continuano a dare una motivazione errata per la stocasticità della matrice del primo esercizio ovvero scrivono: *'La matrice è stocastica perché non ha nessuna colonna di zeri'*. Gli altri tre studenti non scrivono per quale motivo la prima matrice sia stocastica.
Come mi aspettavo, la terza categoria (*Solo il secondo corretto*) è vuota, ovvero nessun ragazzo di entrambe le classi ha sbagliato il primo esercizio ma ha svolto correttamente il secondo.
La quarta categoria, composta da quegli studenti che hanno sbagliato entrambi gli esercizi, è fortunatamente poco numerosa: ne fanno parte due ragazzi della classe di Pesaro ed uno della classe di San Giovanni in Persiceto. Uno dei due ragazzi della classe di Pesaro ha scritto male la quarta colonna della prima matrice ovvero ha scritto:

$$(\frac{1}{2} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$

invece di

$$(\frac{1}{2} \quad 0 \quad 0 \quad 0 \quad \frac{1}{2} \quad 0)\,.$$

La matrice di adiacenza relativa al secondo esercizio è corretta ma non ha parlato della modifica da fare per rendere stocastica tale matrice. Per quanto riguarda il secondo ragazzo, la matrice di adiacenza che ha scritto relativa al primo esercizio è una matrice 6×5 invece che 6×6 in quanto ha completamente omesso la quarta colonna. Il secondo esercizio è incompleto: la matrice di adiacenza non è finita (ha scritto solo le prime 7 colonne) e quindi non ha neppure detto se sia stocastica e/o irriducibile e non parlato della modifica da fare. Il ragazzo della classe di San Giovanni non ha scritto nessuna delle due matrici di adiacenza né ha minimamente parlato della stocasticità di queste; ha solo scritto che ci sono dei loops nei grafi, evidenziando con la penna dove sono, ed accanto ad entrambi i grafi ha scritto la parola *'riducibile'*. Ciò mi fa capire che il ragazzo ha associato la parola *riducibile* ad un grafo invece che ad una matrice e quindi ha fatto confusione fra i termini *fortemente connesso* ed *irriducibile*. Ricordiamo che, se un grafo è fortemente

connesso, la matrice di adiacenza associata è irriducibile e viceversa, per cui questo studente avrebbe dovuto scrivere *non fortemente connesso* accanto ai grafi invece di *riducibile*.
Dell'ultima categoria, relativa alle prove *Non classificabili*, abbiamo già parlato all'inizio della sezione, precisando la grande differenza fra i compiti dei ragazzi della classe di Pesaro e quelli dei ragazzi della classe di San Giovanni. Gli studenti di Pesaro hanno comunque provato, nel poco tempo che avevano a disposizione, a svolgere gli esercizi: uno studente ha scritto la prima matrice di adiacenza precisando che è stocastica, ma non ha motivato tale risposta né ha parlato di irriducibilità. Il secondo studente ha scritto la prima matrice ma ha commesso degli errori nella lettura del grafo. Al contrario i due studenti della classe di San Giovanni, che avevano 20 minuti di tempo a loro disposizione per svolgere gli esercizi, hanno volutamente lasciato la prova completamente in bianco.

6.2 Analisi seconda prova intermedia

La seconda prova intermedia consisteva nella risoluzione di due esercizi con Matlab: il loro testo si trova nel Capitolo 3 a pagina 69. Per svolgere questi esercizi i ragazzi di entrambe le classi hanno lavorato a coppie; ho proposto il primo alla fine della terza lezione mentre il secondo all'inizio della quarta. Tutti gli esercizi che gli studenti mi hanno consegnato sono corretti perché, durante il loro svolgimento, i ragazzi si aiutavano a vicenda: quando commettevano degli errori di sintassi o vedevano che il risultato ottenuto non era quello atteso chiedevano il mio aiuto o quello dei compagni vicini. Per questo motivo, a differenza della prova precedente, non ho suddiviso le prove dei ragazzi in diverse categorie. Quello che faremo in questa sezione sarà analizzare gli errori più comuni che gli studenti hanno commesso nello svolgimento di entrambi gli esercizi. Iniziamo trattando il primo di questi.
I ragazzi di entrambe le classe hanno avuto difficoltà nella prima parte di questo esercizio, ovvero nell'eliminare la terza riga della matrice e nell'ag-

giungere il vettore (1 1) come ultima colonna. Nella classe di Pesaro solo 2 coppie su 8 hanno soddisfatto queste richieste senza aiuti esterni, mentre nella seconda classe vi è riuscita solo una coppia su 6. Vi sono diversi modi per apportare queste due modifiche alla matrice di partenza A. Due ragazze della classe di San Giovanni hanno utilizzato le seguenti istruzioni:

```
A1=[A(1,:);A(2,:)]
a=[1;1]
B=[A1,a]
```

.

Quello che hanno fatto è stato creare per righe una nuova matrice $A1$: tali righe sono le prime due della matrice A. Poi hanno creato un vettore colonna a ed infine hanno aggiunto questo vettore colonna ad $A1$ chiamando B questa nuova matrice 2×4. Gli errori più comini fra i loro compagni che hanno provato ad usare questa strategia sono stati:

- `A1=[A(1,:),A(2,:)]`
- `A1=(A(1,:);A(2,:))`
- `a=[1,1];B=[A1,a]`

.

Il primo è un errore semantico: quando si vuole costruire una matrice per righe, per separare queste ultime, va utilizzato il **;** perché la virgola (**,**) si usa per separare i vari elementi, quindi se scriviamo `A1=[A(1,:),A(2,:)]` non otteniamo una matrice 2×3 ma un vettore 1×6: (2 3 4 3 4 5).
I restanti due sono errori di sintassi, ovvero l'espressione non è scritta correttamente: Matlab non capisce la sequenza, non sa come interpretare il comando e quindi non riesce ad eseguirlo. Il secondo errore riguarda la costruzione di matrici: in Matlab, per creare una matrice, si utilizzano le parentesi quadre [ ]; le tonde **( )** servono per estrapolare elementi dalla matrice. Il terzo errore riguarda l'aggiunta del vettore (1 1) come ultima colonna della matrice $A1$ ed è stato commesso anche da diversi ragazzi della classe di Pesaro. Alcuni studenti hanno costruito tale vettore come un vettore riga e poi, quando hanno cercato di aggiungerlo ad $A1$ come colonna, hanno trovato dei

problemi: non hanno considerato che, avendo creato un vettore riga, nella seconda istruzione dovevano scrivere `a'` (cioè il trasposto di `a`) al posto di `a`. Questo è l'errore che i ragazzi hanno fatto più fatica ad individuare.
La maggior parte degli studenti di Pesaro ha utilizzato un'istruzione diversa e più sintetica per creare la matrice $A1$, ovvero

```
A1=A(1:2,:)
```
.

Usando questo comando si prendono le prime due righe della matrice A e tutte le colonne, quindi non si considera la terza riga della matrice. Un errore di sintassi molto comune associato a questa istruzione è stato:

```
A1=A(1:2;:)
```

ovvero al posto della virgola per separare gli elementi alcuni studenti hanno messo il punto e virgola (;). Infine circa la metà dei ragazzi di entrambe le classi non ha prestato la dovuta attenzione all'ultima richiesta: calcolare la somma degli elementi della matrice C per righe. Questi studenti hanno usato la seguente istruzione:

```
s=sum(C)
```

e, in questo modo, hanno calcolato la somma degli elementi della matrice per colonne. È corretto utilizzare la funzione *sum* per determinare tale somma ma, se le passiamo C come parametro, essa ci fornisce la somma degli elementi della matrice per colonne quindi, per avere la somma degli elementi per righe, bisogna passarle `C'` (la trasposta di C) come parametro. Nel costruire la matrice C e nel determinare il massimo ed il minimo di ciascuna sua colonna i ragazzi di entrambe le classi non hanno avuto problemi.
Per quanto riguarda il secondo esercizio della prova, ricordiamo che i ragazzi della classe di San Giovanni sono stati maggiormente aiutati nel suo svolgimento in quanto avevano avuto maggiori difficoltà iniziali con Matlab. Quando ho proposto questo secondo esercizio era presente quasi tutta la classe, per un totale di 10 coppie (una delle quali era un trio). Dopo aver dato loro degli accorgimenti iniziali (di cui ho parlato nel Capitolo 5 a pagina 101)

la metà degli studenti di tale classe è stata in grado di scrivire il codice senza un mio ulteriore aiuto, ricordandosi anche di mettere *end* alla fine della condizione *if*. Alcuni errori commessi dagli altri ragazzi sono:

- ```
  y=2;
  r=rem(n,y);
  if r=0
  ```
- ```
  rem(n,2)
  if ==0
  ```
- ```
  else A!=0   (dove A=rem(n,2))
  ```

Il primo errore è un errore di sintassi dato dall'inesperienza di questi ragazzi nell'utilizzo di Matlab. Questa coppia di studenti ha correttamente richiamato la funzione *rem* ma si è dimenticata che le condizioni dell'istruzione *if* sono condizioni logiche e quindi, per un'uguaglianza, bisogna inserire un doppio uguale per cui l'istruzione corretta è: `if r==0`. Il secondo errore è più grave in quanto da esso emerge che questa coppia non ha capito il funzionamento del comando *if*. Questi ragazzi hanno richiamato la funzione *rem* ma non all'interno dell'istruzione *if* e non hanno neanche memorizzato tale valore in una variabile. Accanto ad *if* non hanno scritto una vera e propria condizione perché manca completamente il primo termine dell'uguaglianza. Il terzo errore non è particolarmente grave: questi studenti pensavano di dover esplicitare una condizione anche accanto al comando *else*, condizione che non solo non è necessaria (perché Matlab, tutte le volte che non è rispettata la condizione dell'*if*, esegue le istruzioni scritte dopo il comando *else*) ma se scritta Matlab segnala errore. Notiamo comunque che tale condizione è scritta correttamente quindi in un altro contesto non avrebbe generato problemi di sintassi. Infine due ragazze, che avevano scritto correttamente il codice, pensavano che ci fosse qualche errore perché il risultato che ottenevano era sempre lo stesso: 4. In realtà non si ricordavano come lanciare lo script da Matlab: scrivevano sempre 4 e davano invio, non richiamando il nome dello script e cosí, ovviamente, non veniva eseguita alcuna istruzione.
  ```

Analizziamo ora i codici scritti dagli studenti della classe di Pesaro. Questi ragazzi hanno avuto meno suggerimenti su come svolgere il secondo esercizio. Ho spiegato loro il funzionamento di *rem* ma non ho precisato quale fosse il secondo parametro da passarle in questo particolare esercizio. Le varie coppie (8 in totale) si sono aiutate molto fra loro, più di quanto abbiamo fatto le coppie di studenti della classe di San Giovanni. In breve tempo tutti hanno capito che questo secondo parametro doveva essere il numero naturale 2. Ricordiamo che in questa classe erano presenti due studenti, Andrea e Stefano, che sapevano già programmare quindi i loro codici sono leggermente diversi da quelli dei compagni. Andrea, per esempio, non ha utilizzato la funzione *disp* per stampare a video ma la funzione *sprintf*. Dopo aver memorizzato in una variabile m il resto della divisione di n per 2, dove n è il generico naturale passato in input, il ragazzo ha scritto le seguenti istruzioni:

```
if(m == 0)
    s = 'pari';
else
    s = 'dispari';
end
sprintf('Il numero %d  %s', n, s)
```

Quello che ha fatto è stato creare una variabile s in cui è memorizzata una stringa: se m è zero s contiene la stringa *pari*, mentre se m è diverso da zero s contiene la striga *dispari*. Infine, tramite il comando *sprintf*, viene stampata a video la stringa *'Il numero n è pari (dispari)'*, ovvero se, per esempio, $n = 6$ viene stampata la frase *'Il numero 6 è pari.'*. Altre due coppie, oltre quella di Andrea e del suo compagno, hanno usato il comando *sprintf* chiedendo ad Andrea come utilizzarlo. Stefano ed il suo compagno, invece, non hanno utilizzato nè la funzione *disp* nè la funzione *sprintf* e, nel loro codice, hanno specificato che il numero passato come input deve essere un intero positivo affinché si possa stabilire se sia pari o dispari. Lo script creato da questi due studenti è il seguente:

```
n=input('Inserisci il numero: ');
if (n<0)
    'il numero  negativo'
    break;
end
resto = rem(n,2);
if (resto == 0)
    'Il numero  pari'
else
    'Il numero  dispari'
end
```

Dopo aver preso in input il numero n, i ragazzi controllano se tale numero sia < 0. In questo caso, stampano a video la stringa *'il numero è negativo'* ed utilizzano il comando *break* che interrompe immediatamente l'esecuzione del programma così l'operatività torna alla command window (cioè compare un nuovo prompt di comando). Se $n > 0$ viene calcolato il resto della divisione di n per 2 e, se tale resto è uguale a zero, verrà stampata a video la stringa *'Il numero è pari'* mentre se tale resto è diverso da zero la stringa che verrà stampata sarà *'Il numero è dispari'*. Per stampare a video, i ragazzi hanno sfruttato una particolare caratteristica di Matlab: se dopo una qualsiasi istruzione non viene messo il punto e virgola (**;**) il risultato di tale istruzione viene automaticamente stampato a video. Da un punto di vista estetico, questo stratagemma non è molto elegante per il nostro esercizio perché, ogni volta che viene stampata a video una frase, come prima cosa compare `ans =` e di seguito la frase, in quanto le varie stringhe vengono memorizzate nella variabile di default *ans* dato che non sono state memorizzate in una specifica variabile. Le altre 4 coppie di studenti non hanno fatto aggiunte al codice ed hanno utilizzato la function *disp* per stampare a video. Anche in questa classe alcuni ragazzi hanno specificato la condizione accanto al comando *else* ovvero hanno commesso l'errore

```
else rem(n,2)!=0      .
```

6.3 Analisi questionario

L'ultima prova che ho sottoposto alle due classi è stata il questionario di valutazione finale. Ricordiamo che esso è composto da 12 domande ed è diviso in due parti: la prima parte, costituita dalle prime 6 domande, chiede ai ragazzi le loro opinioni sul corso, mentre la seconda, formata dalle restanti 6, è volta a testare le competenze acquisite da questi ultimi. Analizziamo le risposte che i ragazzi della classe di Pesaro hanno dato alle domande del questionario. Le due tabelle sottostanti mostrano come questi studenti abbiano risposto alla prima parte del questionario ovvero alle prime 6 domande. La prima tabella è relativa alle risposte fornite alle domande a scelta multipla, la seconda alle risposte fornite alle domande aperte.

Classe 1 - Totale questionari: 16				
Questionario	Sì	Abbastanza	No	Non so
Domanda 1	15	1	0	0
Domanda 2	12	4	0	0
Domanda 5	13	3	0	0
	troppo	giusto	troppo poco	Non so
Domanda 6	0	10	5	1

Classe 1 - Totale questionari: 16					
Questionario	Lezione 1	Lezione 2	Lezione 3	Lezione 4	Non so
Domanda 3	5	2	4	5	0
Domanda 4	2	6	2	4	2

Come si vede dalla prima tabella, tutte le risposte alle domande 1 e 2 sono positive per cui gli studenti hanno trovato interessante l'argomento del corso ed hanno apprezzato come è stato svolto. Inoltre, come si evince dalle risposte alla domanda 5, a tutti gli studenti della classe è piaciuto lavorare con Matlab. Notiamo che, in entrambe le tabelle, ho aggiunto la colonna *Non so*, anche se questa possibilità non era presente nelle domande a scelta multipla del questionario. Nella prima tabella ho aggiunto questa colonna perché una

ragazza, nel rispondere alla domanda 6, ha marcato due scelte (*giusto* e *troppo poco*) dando la seguente motivazione: *'Non posso saperlo perché non so quante cose non abbiamo fatto'*. Siccome tale risposta mi è sembrata degna di nota ho apportato questa aggiunta alla tabella. Osserviamo che le risposte alla domanda 6 sono meno omogenee rispetto alle risposte relative alle precedenti domande a scelta multipla, infatti emerge che un terzo della classe ritiene che il tempo dedicato a Matlab sia stato poco. Anche nella seconda tabella, inerente alle domande 3 e 4, è presente la colonna *Non so*, in quanto due studenti non hanno saputo dire quale fosse la lezione rimastagli meno impressa. Uno di questi due studenti scrive: *'Nessuna in particolare. Mi hanno complessivamente interessato tutte'*. Osservando le altre risposte alla domanda 4, notiamo che la seconda lezione è stata quella che è piaciuta di meno agli studenti. I sei ragazzi hanno dato due tipi di motivazioni per questa loro scelta: tre studenti hanno trovato la lezione molto teorica mentre per i restanti tre la lezione ha trattato argomenti difficili. Riporto due risposte appartenenti ai due diversi gruppi rispettivamente:
'La seconda perché era tutta teorica e le definizioni (stocastica, ben posta,...) non mi sono rimaste in mente'.
'La seconda perché le definizioni di alcuni elementi e di alcune variabili mi sono state difficili da capire'.
Quattro studenti hanno scritto che la quarta lezione è quella che è rimasta loro meno impressa dando tutti la stessa motivazione: il tempo per analizzare l'algoritmo è stato poco e quindi, per loro, la lezione è stata superficiale. Uno di questi ragazzi scrive: *'L'ultima perché abbiamo avuto poco tempo per capire cosa facesse di preciso il codice per ottenere effettivamente il Pagerank'*. Per quanto riguarda la domanda 3, le lezioni 1 e 4 sono, a pari merito, le lezioni rimaste più impresse. Fra i cinque ragazzi che hanno scelto la prima lezione, due scrivono: *'La prima perchè abbiamo mosso i primi passi all'interno di argomenti nuovi e suggestivi'* e *'La prima è stata la più interessante perché non conoscevo per niente l'argomento relativo al Pagerank'*, quindi la prima lezione è quella rimasta loro più impressa perchè hanno visto per

la prima volta il motore di ricerca Google sotto un nuovo punto di vista. Gli altri tre ragazzi invece motivano la loro scelta in modi differenti. Un ragazzo scrive: *'La prima lezione quando abbiamo parlato del Pagerank di una pagina e del sistema matriciale che utilizza Google perché, essendo uno che aveva un forum su Internet, mi interessavano i discorsi sul rango e gli outlinks'*. Una ragazza ha trovato interessante il sistema di ricerca originario di Google, mentre al terzo ragazzo è piaciuta molto *'la spiegazione su come Google elenca i risultati trovati.'* Quattro dei cinque ragazzi che hanno scelto la quarta lezione scrivono che questa è la lezione rimasta loro più impressa perché hanno visto concretamente l'algoritmo di indicizzazione di Google. Uno studente invece dá una motivazione molto differente: *'L'ultima perché abbiamo usato Matlab quando già conoscevo abbastanza cose su quel programma'*. È interessante notare la grande differenza di opinioni in merito alla lezione 4: per cinque studenti questa è stata la lezione più interessante mentre per altri quattro (quasi lo stesso numero) è stata la lezione peggiore. Ora trattiamo la seconda parte del questionario per indagare quali concetti sono rimasti in mente ai ragazzi. La domanda 7 chiedeva di dare una definizione di Pagerank. Ho suddiviso le risposte in 5 gruppi in base alle parole maggiormente utilizzate dagli studenti nelle loro risposte. Non mi è sembrato adeguato fare una suddivisione del tipo giusto/sbagliato perché tutti gli studenti hanno utilizzato un linguaggio molto personale e quindi, dal punto di vista matematico, tutte le risposte sarebbero state non pienamente corrette. La seguente tabella riassume le varie tipologie di risposte:

Classe 1 - Totale questionari: 16					
Questionario	Formula	Importanza	Algoritmo	Rango	Numero
Domanda 7	3	5	4	2	2

Tre studenti hanno dato la definizione di Pagerank tramite la formula ricorsiva vista la prima lezione e uno di questi ha anche scritto quella come problema agli autovalori. Cinque studenti hanno definito il Pagerank come l'importanza di una pagina web. C'è chi ha scritto semplicemente *'Indica quanto una pagina è importante'* e chi ha dato definizioni più approfondite

come la seguente: *'Il Pagerank è il grado di importanza di una data pagina web e può variare da 1 a 10. È dato prevalentemente dagli* inlink *che arrivano alla pagina web e dalla rispettiva importanza da cui questi* link *partono'.* Quest'ultimo ragazzo, quando ha scritto la scala dei valori che il Pagerank può assumere, ha pensato alla scala della Google Toolbar di cui avevamo parlato nelle prime lezioni. Un tipo di risposta a cui non avevo minimamente pensato è stata quella data dai quattro ragazzi della categoria *Algoritmo.* Questi studenti alla parola Pagerank hanno associato l'algoritmo Google's Pagerank visto l'ultima lezione e quindi hanno cercato di spiegare cosa fa questo algoritmo. Un ragazzo scrive *'È il metodo efficace con cui Google riesce a classificare per importanza le pagine che compaiono dopo una data ricerca ...'*, mentre per un'altra ragazza *'Il Pagerank è un algoritmo che permette di osservare e calcolare il rango di un insieme di pagine web, ...'.*
Due studenti invece hanno utilizzato la parola *rango* nelle loro risposte, mentre altri due hanno utilizzato la parola *numero.* Enunciamo due risposte appartenenti alle due rispettive categorie:
'Il Pagerank indica il rango di una pagina web cioé prende in considerazione sia gli inlink *che gli* outlink *e in base al numero delle pagine che il sito punta e da cui è puntato si determina la sua importanza'.*
'Il Pagerank è un numero che viene associato al sito (in realtà ad ogni pagina), che varia in base al numero di inlink *e* outlink *del sito. Viene definito ricorsivamente e aumenta con l'aumentare di* link *di altri siti verso quello preso in considerazione'.*
Per quanto riguarda la domanda 8, essa è molto simile, come tipologia, agli esercizi somministrati nella prima prova intermedia e notiamo che è costituita da tre domande interne. In questo caso, le categorie in cui ho suddiviso le risposte sono tre: *Corretto*, di cui fanno parte quelle risposte che sono senza errori in tutte le loro parti ovvero gli studenti che hanno risposto correttamente a tutte le domande interne, *Parzialmente corretto*, categoria a cui appartengono quegli studenti che hanno risposto correttamente a due delle tre domande interne, e *Sbagliato*, categoria costituita dagli studenti che han-

no risposto correttamente solo ad una o a nessuna delle tre domande. La tabella relativa all'ottava domanda è la seguente:

Classe 1 - Totale questionari: 16			
Questionario	Corretto	Parzialmente corretto	Sbagliato
Domanda 8	5	10	1

Da essa emerge che un terzo della classe ha risposto correttamente a tutte e tre le domande interne ma più della metà della classe rientra nella categoria *Parzialmente corretto*. Sette di questi dieci ragazzi non hanno risposto correttamente alla terza domanda che chiedeva come modificare la matrice per risolvere il problema creato della colonna di tutti zeri: quattro ragazzi non hanno proprio risposto a questa domanda, mentre gli altri tre hanno dato risposte sbagliate, del tipo *'sostituisco alla terza colonna una colonna di tutti 1'* oppure *'sommo la colonna 2 alla 3'*.
Con mio grande stupore, quasi tutti gli studenti hanno risposto correttamente alla domanda 9 ovvero hanno correttamente disegnato il grafo relativo alla matrice di adiacenza proposta nell'esercizio. Solo un ragazzo ha sbagliato a disegnare il grafo, tutti gli altri quindici studenti lo hanno scritto correttamente. Io pensavo che questo esercizio potesse essere più problematico di altri dato che, fino a quel momento, avevamo sempre trattato il caso contrario ovvero partendo dal grafo avevamo costruito la matrice di adiacenza, invece per gli studenti questa nona domanda è stata la più facile.
Per quanto riguarda la domanda 10, le categorie in cui ho suddiviso le risposte dei ragazzi sono: *Inlinks, Outlinks, Sia inlinks sia Outlinks*. Come per la domanda 7, ho creato queste categorie in base alle parole ed ai concetti più ricorrenti nelle risposte dei ragazzi e non in base al fatto che tali risposte siano giuste o sbagliate. Ecco la tabella riassuntiva:

Classe 1 - Totale questionari: 16			
Questionario	Inlinks	Outlinks	Sia Inlinks sia Outlinks
Domanda 10	7	4	5

La categoria *Inlinks* è costituita da quelle risposte in cui l'idea principale che emerge, per ottimizzare il ranking di un sito, è di aumentare il più possibile il numero di *inlinks*. Uno di questi sette ragazzi precisa anche che è bene scegliere, come *inlinks*, pagine che abbiamo un rango alto ed inoltre suggerisce di rendere il sito molto accattivante. La sua risposta è la seguente: *'Innanzi tutto consiglierei di creare il sito in modo da catturare l'attenzione del visitatore che poi potrebbe tornare. Inoltre creare molti* inlinks *in pagine con un Pagerank alto.'* Un'altro studente giustamente precisa di indirizzare questi *links* alla *home page* del sito: *'Gli consiglierei di fare in modo che più pagine possibili appartenenti a siti altrui puntino alla homep page del suo sito.'* Nelle quattro risposte appartenenti alla categoria *Outlinks*, gli studenti consigliano di aumentare il numero degli *outlinks* ed uno di loro precisa che, in questo modo, il rango della pagina aumenterà: *'Gli consiglierei di ottimizzare la sua pagina aggiungendo* outlinks *perché in questo modo il rango della pagina aumenterá'.* Purtroppo un tale suggerimento è sconsigliabile perché, aumentando gli *outlinks*, si disperde il rango della nostra pagina web quindi la motivazione data nella risposta precedente è sbagliata. A mio parere questi quattro ragazzi hanno fatto confusione fra i concetti di *inlink* e *outlink*. Le restanti cinque risposte appartengono alla categoria *Sia inlinks sia outlinks*, ovvero in queste risposte appaiono entrambi i concetti. Tutti i cinque ragazzi consigliano di avere molti *inlinks* verso la propria pagina ma hanno posizioni differenti riguardo al numero di *outlinks* da inserire: tre studenti suggeriscono al loro ipotetico amico di *'non avere molti outlinks'* o di *'ridurre il numero di outlinks dalla sua pagina'*, mentre gli altri due studenti consigliano di *'aggiungere più outlinks'*. Uno di questi due studenti però fa una precisazione: *'Potrebbe creare outlinks dalla sua pagina verso altre pagine web importanti che però abbiamo qualcosa in comune con la sua pagina (sennò avrà più una perdita).'* Questo consiglio è ragionevole, in quanto è opportuno avere qualche *link* in uscita dal proprio sito che indirizzi verso pagine in cui si trovano possibili approfondimenti o aggiornamenti. In questo modo il sito acquista anche autorevolezza non avendo l'assurda pretesa

di poter soddisfare ogni singola richiesta dell'utente. Infine notiamo che tre studenti (su sedici) consigliano anche di avere della pubblicità nel sito, ma in giusta quantità, ed una ragazza suggerisce di *'confrontare il proprio sito con quelli che trattano gli stessi argomenti'*.

La domanda 11 è stata quella in cui la varietà delle risposte è stata maggiore. Quindici studenti hanno utilizzato la parola *programma* per definire Matlab ed uno ha utilizzato la parola *software*. Anche per questa domanda le suddivisioni che ho creato variano in base alle espressioni utilizzate dai ragazzi nelle loro risposte ma, a differenza della domande precedenti, queste ripartizioni non sono esclusive perché i ragazzi hanno usato varie espressioni per descrivere Matlab. Per sette di loro Matlab è un programma che ha la matrice come struttura dati di base; due di questi studenti precisano anche l'origine dalla parola Matlab ovvero scrivono: *'Matlab = laboratorio di matrici'*. Alcune risposte date dagli studenti sono: *'Matlab è un programma che utilizza un linguaggio matriciale per programmare'*, *'Matlab è un programma scientifico per agevolare calcoli di matrici e matematici'* e *'Matlab è un programma per eseguire calcoli basato sulle matrici'*. Notiamo che la studentessa che ha scritto quest'ultima risposta precisa anche che Matlab esegue calcoli e, oltre a lei, altri sei ragazzi sottolineano questo fatto. Due di loro scrivono: *'È un programma che svolge calcoli matematici e funzioni'* e *'È un programma che calcola espressioni matematiche a algoritmi'*. Tre ragazzi precisano che Matlab ha un proprio linguaggio di programmazione e solo uno (fortunatamente) scrive: *'Matlab è un programma che funge da calcolatrice'*. Molti ragazzi (sette precisamente) sottolineano che in Matlab è possibile utilizzare funzioni già implementate e scriverne di nuove. Uno studente scrive *'È un programma per sfruttare tantissime funzioni che permettono i migliori usi'*, mentre la risposta di un'altro ragazzo termina con *'... Inoltre è possibile scrivere proprie funzioni o script'*. Come ultima cosa, notiamo che due studenti citano l'algoritmo Pagerank ovvero scrivono che con Matlab abbiamo implementato tale algoritmo. Una risposta in questo senso dice: *'Matlab è un programma con il quale possiamo svolgere numerose*

funzioni matematiche, ad esempio ci può aiutare a calcolare il Pagerank di un sito web'.

Passiamo ora all'ultima domanda del questionario: la domanda 12. Purtroppo diversi studenti sono stati intimiditi da questa domanda, infatti ben cinque ragazzi non provano neanche a dare una risposta ovvero cinque domande vengono lasciate totalmente in bianco. Quattro studenti scrivono qualche frase ma sono risposte superficiali del tipo: *'È un'algoritmo che serve per spiegare a Matlab che procedimento deve intraprendere'*, oppure *'L'algoritmo scritto con Matlab serve per spiegare la funzione necessaria per determinare il Pagerank di un insieme di pagine web'.* Un ragazzo che era stato assente la quarta lezione scrive *'Non posso rispondere perché assente alla quarta lezione'*, mentre un altro studente, invece di descrivere l'algoritmo, riporta la definizione ricorsiva di Pagerank. Egli scrive: *'Tramite l'algoritmo si può dare il valore di Pagerank ad una pagina. Per fare ciò è necessario sommare i valori (Pageranks) delle pagine in entrata diviso il numero dei loro outlinks'.* La definizione riportata è corretta ma, purtroppo, non è pertinente alla domanda. È interessante notare che nella domanda 7, dove si chiedeva di definire il Pagerank e quindi dove lo studente avrebbe potuto scrivere quanto letto sopra, egli dà una definizione molto più vaga: *'Un valore numerico variabile associato ad una pagina web, utile a definire il valore e quindi la visibilità, definito tramite gli inlinks e gli outlinks '.* In definitiva, solo cinque studenti cercano di spiegare il codice o, per lo meno, qualche sua riga. Tre di questi studenti hanno specificato che G è la matrice di connettività, una ragazza ha spiegato cosa fanno le funzioni *sum, size e ones*, mentre un'altro studente ha descritto la prima istruzione. Egli scrive: *'... dobbiamo eliminare i numeri diversi da 0 nella diagonale principale perciò gliela sottraiamo per evitare problemi'.* Solo due studenti hanno specificato che, come output, avremo un vettore le cui componenti sono i Pageranks delle pagine web. La risposta che considero più completa è la seguente: *'Il codice calcola il pagerank prendendo come input una matrice G, che sarebbe quella di connettvità. Si effettuano delle operazioni su G, per estrapolare i parametri che ci servono*

(prime quattro righe) e si creano due matrici. Alla fine otterremo il vettore dei pagerank x'.

Ora vediamo come la classe di San Giovanni in Persiceto ha risposto al questionario. Ricordiamo che le lezioni 2 e 3 sono state svolte di pomeriggio e solo metà della classe era presente ed inoltre parte degli studenti presenti la seconda lezione erano poi stati assenti la terza e viceversa, per cui molti studenti non hanno potuto rispondere a delle domande perché erano stati assenti in una o più lezioni. Iniziamo analizzando le risposte che i ragazzi hanno dato alle prime 6 domande del questionario nelle quali veniva chiesto di esprimere le loro opinioni. Le tabelle sottostanti sono analoghe a quelle di pagina 116.

Classe 2 - Totale questionari: 22				
Questionario	Sì	Abbastanza	No	Non so
Domanda 1	17	5	0	0
Domanda 2	11	11	0	0
Domanda 5	6	15	1	0
	troppo	giusto	troppo poco	Non so
Domanda 6	0	16	5	1

Classe 2 - Totale questionari: 22					
Questionario	Lezione 1	Lezione 2	Lezione 3	Lezione 4	Non so
Domanda 3	8	4	1	7	2
Domanda 4	0	2	8	4	8

Dalla prima tabella emerge che anche questa classe ha trovato interessante l'argomento del progetto e, complessivamente, ha apprezzato come è stato svolto il corso. Come mi aspettavo, Matlab è piaciuto di meno a questi studenti, infatti solo cinque hanno risposto *Sì* alla domanda 5 ed uno ha barrato la casella *No*. Però, per quanto riguarda il tempo dedicato all'utilizzo di Matlab, la maggior parte dei ragazzi pensa che sia stato adeguato. Le lezioni che sono piaciute di più a questi studenti sono state la prima e la quarta. Fra i sette ragazzi che hanno scelto l'ultima lezione, uno non fornisce spiegazioni,

una ragazza scrive *'Mi è rimasta più impressa la quarta perché abbiamo visto come applicare tutto ciò di cui avevamo studiato teoricamente nelle lezioni precedenti'*, mentre i restanti cinque forniscono tutti la stessa motivazione: ciò che li ha entusiasmati è stato vedere e testare con le loro mani l'algoritmo Google's Pagerank. Gli otto ragazzi che, invece, hanno scelto la prima lezione danno motivazioni molto differenti: a due studenti è piaciuta la prima lezione perché è stato introdotto un argomento *'nuovo ed interessante'*; uno studente ha preferito la prima lezione alle altre perché questa è stata quella che ha capito maggiormente, mentre una studentessa scrive: *'La prima perché è stato spiegato come funziona l'ordinamento della pagine in Google e la costruzione delle matrici'*. Ai restanti quattro è rimasta particolarmente impressa la prima lezione perché hanno trovato interessante la spiegazione generale del funzionamento di Google. Due di questi studenti scrivono: *'La prima perché abbiamo analizzato il funzionamento di Google'* e *'La prima per la spiegazione generale dell'algoritmo di Google'*.

Purtroppo già nelle risposte alla domanda 4 il numero di *Non so* inizia ad essere alto. Sette risposte che appartengono a questa categoria sono del tipo: *'La seconda, perché non c'ero'*, *'La prima perché ero assente'* oppure *'Purtroppo ho partecipato a due sole lezioni'*. Solo un ragazzo ha lasciato la risposta completamente in bianco. Altri otto studenti hanno scritto che la lezione che è rimasta loro meno impressa è, come mi aspettavo, la lezione 3. Due ragazzi motivano la loro scelta in questo modo: *'La terza perché era troppo difficile'* e *'La terza perché non non mi è venuto quasi niente'*. Un altro studente scrive *'La terza perché abbiamo fatto solo esercizi'*, mentre una ragazza sottolinea che, come classe, non avevano mai programmato dicendo *'Mi è rimasta meno impressa la prima lezione in cui abbiamo utilizzato Matlab perché non siamo abituati ad utilizzare programmi di programmazione al pc'*. Le altre motivazione date per questa scelta sono molto simili a quelle appena viste.

Ora passiamo alle seconda parte del questionario. Per quanto riguarda la domanda 7, le categorie in cui ho suddiviso le varie risposte sono le stesse

viste per classe di Pesaro con l'aggiunta della categoria *Non so*. La tabella riassuntiva è la seguente.

Classe 2 - Totale questionari: 22						
Questionario	Formula	Importanza	Algoritmo	Rango	Numero	Non so
Domanda 7	0	7	8	1	2	4

Nelle quattro risposte appartenenti alla categoria *Non so* i ragazzi scrivono *'Non c'ero'* ma non precisano a quale lezione. Con mia sorpresa ben otto ragazzi associano alla parola Pagerank la parola algoritmo. Due studenti fornisco la stessa identica risposta: *'È un algoritmo che stabilisce l'importanza di un collegamento in una serie"*, mentre un ragazzo scrive *'È un algoritmo che serve a stabilire l'ordine di importanza di una pagina'*. Come si può constatare da questa risposte, in molte compare anche la parola importanza. Una ragazza fornisce una risposta davvero particolare: *'È un insieme di algoritmi'*. Ho riflettuto molto sul perché questa ragazza abbia usato il plurale e credo che lei, di fronte alla parola Pagerank, abbia pensato ai due algoritmi visti nell'ultima lezione, il primo dei quali determina il vettore Pagerank x attraverso il metodo delle potenze mentre il secondo risolve un sistema lineare, quindi ha parlato di *'insieme di algoritmi'* perché, effettivamente, ne ha visti due che portano allo stesso risultato. Sette ragazzi hanno parlato di importanza. C'è chi ha scritto semplicemente *'Il grado di importanza di una pagina'* e chi ha fornito spiegazioni più approfondite: *'Il pagerank è l'importanza che ha una pagina web e viene definita attraverso i link in entrata ed in uscita nella pagina stessa e da dove questi link arrivano'*. Tre studenti hanno dato risposte quasi identiche, per cui ne riporto solo una: *'Il pagerank indica l'importanza di una pagina all'interno del web. Dipende da quanti link rimandano ad una determinata pagina'*. Nessun studente ha riportato la formula vista la prima lezione ed i due ragazzi che hanno utilizzato il sostantivo *numero* hanno dato la stessa identica risposta, purtroppo sbagliata: *'Il Pagerank è il numero di outlinks di una pagina'*.

Anche per la domanda 10 le categorie in cui ho suddiviso le varie risposte sono le stesse viste per la classe precedente con l'aggiunta della categoria *Non*

so. Ecco le tabella riassuntiva:

Classe 2 - Totale questionari: 22				
Questionario	Corretto	Parzialmente corretto	Sbagliato	Non so
Domanda 8	10	10	1	1

Circa la metà della classe ha risposto correttamente a tutte e tre le domande interne, solo una risposta rientra nella categoria *Sbagliato* (ovvero questo studente ha risposto solo ad una delle tre domande interne) e solo un ragazzo ha lasciato la risposta completamente in bianco, non precisando se era stato assente o meno a qualche lezione. La maggior parte degli studenti le cui risposte rientrano nella categoria *Parzialmente corretto* (precisamente sei studenti) hanno risposto correttamente alla prima ed alla terza domanda interna, ovvero hano saputo individuare quale fosse la colonna problematica ed hanno presentato una corretta modifica di tale colonna per risolvere il problema, ma non non hanno saputo spiegare in cosa consistesse tale problema. Quattro di questi sei ragazzi non hanno fornito alcuna risposta alla seconda domanda mentre due ragazzi hanno dato una risposta sbagliata. Essi hanno scritto *'La terza colonna, perché è di soli zeri e perció non ci sono link che rimandano a quella pagina'* e *'La terza è problematica perché è creato un loop'*. Il fatto che la terza colonna sia di tutti zeri non vuol dire né che vi è un loop nel grafo né che la pagina non ha *inlinks* ma significa che tale pagina non ha *links* in uscita, ovvero *outlinks*, per cui si rimane bloccati. I restanti quattro ragazzi non rispondono correttamente alla terza domanda interna; più precisamente nessuno di loro tenta di rispondere a questa domanda. È interessante notare che due di questi quattro ragazzi rispondono alla seconda domanda in modo molto diverso rispetto a tutti gli altri loro compagni ovvero scrivono: *'La terza colonna perché la somma dei suoi elementi non è 1'*. Anche questa risposta è corretta perché, per poter calcolare i Pageranks di un insieme di pagine web, la matrice di adiacenza deve essere, per prima cosa, stocastica (ed irriducibile).

Come per la classe di Pesaro, la domanda 9 è stata quella che gli studenti hanno considerato più facile. Ben diciotto di loro hanno disegnato corretta-

mente il grafo relativo alla matrice di adiacenza proposta e solo due studenti non lo hanno disegnato correttamente. I restanti due ragazzi hanno lasciato la domanda in bianco non precisando se fossero stati assenti o meno alla prima o seconda lezione.
Nella domanda 10 il numero di *Non so* torna ad aumentare: quattro ragazzi lasciano la domanda in bianco e due scrivono *'Non c'ero'*. Ecco la tabella riassuntiva della risposte:

Classe 2 - Totale questionari: 22				
Questionario	Inlinks	Outlinks	Home page	Non so
Domanda 10	9	2	5	6

Come si vede dalla tabella, le categorie in cui ho suddiviso le varie risposte sono leggermente diverse da quelle create per la classe precedente, infatti oltre alle categorie *Inlinks* e *Outlinks*, costituite da quelle risposte in cui gli studenti parlano principalmente della gestione di *inlinks* e *outlinks* rispettivamente, vi è la nuova categoria *Home page* costituita da quelle risposte in cui gli studenti consigliano come gestire la *home page*. Anche in questa classe la categoria *Inlinks* è molto numerosa e tutti questi nove studenti consigliano di aumentare il più possibile il numero di *inlinks*. Al contrario la categoria *Outlinks* è costituita solo da due studenti: uno di questi consiglia di *'evitare troppi outlink'* mentre, erroneamente, il secondo studente propone il consiglio contrario: *'Di mandare outlinks dalla sua pagina a pagine con pagerank maggiore'*. Quest'ultimo ragazzo ha sicuramente fatto confusione fra i concetti di *inlink* e *outlink*. Per quanto riguarda la nuova categoria, in tutte queste cinque risposte i ragazzi forniscono due principali consigli, entrambi corretti: non inserire *outlinks* nella *home page* ma creare una pagina specifica che li contenga tutti. Una ragazza scrive: *'Gli consiglio di lasciare la home page priva di link e di inserire una pagina apposta con tutti i collegamenti alle altre pagine'*. Infine notiamo che uno studente consiglia anche di *'rendere piacevole dal punto di vista grafico le pagina e inserire in essa parole chiave'*, mentre una ragazza sottolinea *'di rendere la propria pagina più interessante possibile'*, ovvero di dare molta importanza ai contenuti.

Per quanto riguarda la domanda 11, purtroppo le risposte date da questi studenti sono molto più sintetiche rispetto a quelle fornite dai ragazzi della classe di Pesaro, basta notare che cinque studenti, per spiegare cosa sia Matlab, scrivono semplicemente *'È un programma'*. A questa domanda hanno risposto tutti i 22 studenti, quindi nessuno ha lasciato la risposta in bianco o ha precisato che era stato assente a qualche lezione. Inoltre, ben 21 studenti hanno utilizzato la parola *'programma'* per definire Matlab. Come nella classe precedente, alcuni ragazzi (cinque) hanno precisato che la matrice è la struttura base di Matlab. Un ragazzo scrive: *'Matlab è un programma per il calcolo numerico che si basa sulle matrici'*, mentre altri studenti forniscono risposte meno dettagliate del tipo *'Matlab è un programma informatico che ti premette di lavorare agevolmente con le matrici'*. I restanti undici studenti parlano della possibilità di poter utilizzare funzioni già implementate in Matlab e di poterne scrivere di nuove. Notiamo che ben otto ragazzi su undici utilizzano la parola *algoritmo* al posto di funzione. Riporto tre risposte date dagli studenti:

'Matlab è un programma con cui si possono creare algoritmi per risolvere determinati problemi.'

'È un programma per lo sviluppo di algoritmi.'

'Matlab è un programma per la programmazione in cui puoi utilizzare varie funzioni matematiche'.

Solo un ragazzo in tuttla classe non ha utilizzato la parola *programma* per definire Matlab ma ha identificato quest'ultimo con il suo linguaggio di programmazione; egli scrive: *'Matlab è un linguaggio di programmazione'*. Questa risposta non è corretta perché Matlab ha (non è) un proprio linguaggio di programmazione e quindi andava specificato che è un ambiente interattivo.

Passiamo ora all'ultima domanda del questionario: la domanda 12. Purtroppo, a differenza della domanda precedente, molti studenti non hanno risposto a quest'ultima: le risposte lasciate totalmente in bianco sono state quattro, mentre tre studenti hanno scritto di essere stai assenti alla quarta lezione. Ben dieci studenti provano a rispondere alla domanda ma danno delle spie-

gazioni molto vaghe e superficiali. Quattro di questi dieci ragazzi forniscono la stessa risposta: *'Il codice permette di trovare il pagerank di ogni pagina che si trova nella matrice G'*. Gli altri sei studenti forniscono risposte diverse da questa ma dai contenuti molto simili. I restanti cinque studenti provano seriamente a spiegare le varie righe di codice; due di questi spiegano anche come viene determinato il vettore Pagerank x: attraverso la risoluzione di un sistema lineare. Ricordiamo che nessuno degli studenti della classe di Pesaro aveva fatto questa importante osservazione. Uno di questi due studenti dà anche informazioni sull'ultima riga di codice; riporto la risposta fornita da questa ragazza: *'Dopo aver determinato la matrice D e la matrice identità I risolviamo il sistema lineare uguagliandolo ad e, così calcoliamo il vettore pagerank x; infine dividiamo x per un altro valore per semplificare i risultati, con un passaggio facoltativo'*. Infine uno studente, pur non specificando che verrà risolto un sistema lineare, fornisce la seguente ottima spiegazione: *'Il codice ci permette di trovare il pagerank di ogni pagina che è inserita in una matrice G data. Definite le variabili c, r, k, e, prendiamo una matrice identità I e attraverso l'equazione* $x = (I - p \cdot G \cdot D) \backslash e$ *troviamo il ranking di ogni pagina'*.

6.4 Conclusioni finali

Siamo giunti alla fase finale di questo progetto. Fino ad ora lo abbiamo analizzato in ogni sua singola parte, ovvero quello che abbiamo fatto è stato proporre un quadro generale riguardante i concetti trattati e le prove somministrate ai ragazzi, parlare di ogni singola lezione e delle reazioni dei ragazzi ed infine abbiamo analizzato le risposte di questi ultimi alle tre prove somministrate. Ora possiamo dire quali obiettivi sono stati raggiunti dagli studenti, quali sono stati gli aspetti meglio riusciti del progetto e quali sono le principali modifiche da apportare. Ovviamente, le due classi hanno raggiunto obiettivi diversi dato che si tratta di studenti che hanno un passato scolastico e personale differente ma anche perché le condizioni in cui si è

svolto il progetto sono state diverse: con la classe di Pesaro ho avuto a disposizione 8 ore del consueto orario scolastico, quindi le lezioni si sono svolte tutte di mattina; con la classe di San Giovanni in Persiceto, la seconda e terza lezione le ho svolte di pomeriggio, dopo l'orario scolastico, e questo ha inciso negativamente sul modo in cui i ragazzi si sono approcciati a Matlab dato che erano chiaramente stanchi. Un'altra conseguenza negativa legata a questa suddivisione delle lezioni è stata che non tutta la classe era presente alle lezioni pomeridiane e quindi diversi studenti non hanno potuto svolgere tutte le prove. In base a quanto detto, il primo consiglio che mi sento di dare a chi, in futuro, vorrà proporre una simile sperimentazione è scegliere opportunamente gli orari in cui si svolgeranno le lezioni.
Ritornando agli obiettivi (la cui lista si trova a pagina 67), quelli che sono stati raggiunti dai ragazzi di entrambe le classi sono: l'obiettivo 2) e l'obiettivo 3). Per quanto riguarda l'obiettivo 2), il 50% degli studenti che svolto la prima prova intermedia, ovvero 13 ragazzi su 26, ha scritto correttamente le matrici di adiacenza relative ai due grafi proposti nella prova e molti dei ragazzi che non hanno scritto correttamente la matrice relativa al secondo esercizio hanno commesso dei banali errori di distrazione come quello visto a pagina 108. Ancora più soddisfacente è stato il risultato avutosi per la domanda 9 del questionario: l'86 % degli studenti, cioè 33 ragazzi su 38, è stato in grado di disegnare correttamente il grafo relativo alla matrice di adiacenza proposta. Per quanto riguarda l'obiettivo 3), il 65% dei ragazzi (17 su 26) individua correttamente sia la stocasticità e riducibilità della prima matrice relativa alla prima prova intermedia sia la non stocasticità e riducibilità della seconda matrice. Ricordiamo che tutti questi ragazzi danno una motivazione corretta della riducibilità delle due matrici mentre hanno più problemi a giustificare correttamente la stocasticità della prima matrice. Nel questionario, 26 ragazzi su 38, ovvero il 68%, hanno saputo correttamente spiegare quale fosse il problema della matrice relativa alla domanda 8, problema legato alla sua non stocasticità.
L'obiettivo 1) e l'obiettivo 7) sono stati raggiunti solo dalla classe di Pesa-

ro. Per quanto riguarda l'obiettivo 1), il 70% delle risposte fornite da questi ragazzi alla domanda 7 del questionario, in cui veniva chiesto di definire il Pagerank, sono coerenti e approfondite a prescindere dal fatto che i ragazzi abbiamo definito il Pagerank come importanza, rango, numero o attraverso la formula. Anche i 4 ragazzi che hanno parlato dell'algoritmo Google's Pagerank hanno fornito risposte coerenti ma non credo che abbiamo capito a fondo la differenza fra questo concetto ed il funzionamento dell'algoritmo. L'algoritmo non coincide con il Pagerank, esso ci permette di determinare il Pagerank di un insieme di pagine web. Nelle risposte di alcuni ragazzi emerge questa differenza, ma in altre non è chiara. Il fatto che diversi studenti (4 della prima classe e 8 della seconda) abbiamo associato alla parola Pagerank la parola algoritmo mi fa capire che, in una futura sperimentazione, sarebbe più opportuno insistere maggiormente su questo concetto per evitare fraintendimenti da parte degli studenti tra il funzionamento dell'algoritmo ed il Pagerank stesso ed impedire così la nascita di misconcezioni. Per quanto riguarda l'obiettivo 7), la classe di Pesaro ha potuto toccare con mano le differenze fra l'algoritmo Google's Pagerank visto a lezione ed i più recenti algoritmi utilizzati da Google attraverso le creazione di un sito web che ha premesso loro di venire direttamente a contatto con la gestione della pubblicità e della *home page*, i contenuti degli articoli, il *linking* interno, ecc...
Al contrario l'obiettivo 4) è stato raggiunto solo dalla classe di San Giovanni. Già nella prima prova intermedia 6 studenti su 11 (il 55%) hanno correttamente modificato la matrice di adiacenza relativa al secondo grafo al fine di renderla stocastica e la percentuale è aumentata nel questionario: il 73% dei ragazzi (16 ragazzi su 22) ha modificato correttamente la matrice proposta nella domanda 7 per renderla stocastica. Per quanto riguarda il procedimento per rendere una matrice irriducibile, ho preferito non proporre esercizi a riguardo in quanto si tratta di una procedura lunga e schematica che avrebbe sottratto tempo a cose più funzionali e di maggior interesse, quali il lavoro con Matlab, per cui, a posteriori, direi di modificare l'obiettivo 4) eliminando l'applicazione di procedimenti per rendere le matrici irriducibili.

Un'ulteriore modifica che apporterei riguarda il quinto obiettivo. Durante il lavoro con Matlab mi sono resa conto che 4 ore erano insufficienti per poter raggiungere un tale obiettivo, dato che quasi la totalità degli studenti non aveva mai programmato. Essi sono riusciti a svolgere gli esercizi con Matlab ma non in modo autonomo, hanno avuto molto bisogno del mio aiuto e del supporto di alcuni compagni. In definitiva, direi che l'obiettivo 5) andrebbe semplificato perché non è possibile raggiungerlo con un lavoro di 4 ore oppure, per chi non lo volesse modificare in una futura ripresentazione di tale progetto, consiglio di aumentare il numero di ore dedicate a Matlab.
L'obiettivo che resta da analizzare è il sesto. Purtroppo, nessuna delle due classi ha raggiunto tale obiettivo e ciò è emerso dalle risposte che i ragazzi hanno fornito alla domanda 12 del questionario. Il 34% degli studenti (13 studenti su 38) non ha nemmeno provato a rispondere a questa domanda, ben 9 studenti l'hanno lasciata in bianco e 4 hanno scritto di essere stati assenti all'ultima lezione. Il 39% ha dato risposte molto vaghe e superficiali, spiegando solo quale fosse l'obiettivo del codice senza entrare in merito a come, cioè attraverso quali operazioni, si raggiungesse tale obiettivo. Il restante 27% tenta di spiegare qualche riga di codice ma solo il 15% (6 ragazzi su 38) danno risposte pienamente soddisfacenti spiegando chi è il vettore x che si ha come output e che, per ottenerlo, si risolve un sistema lineare.
Dall'analisi degli obiettivi e dalle risposte che i ragazzi delle due classi hanno fornito per la prima parte del questionario possiamo concludere che la prima lezione non ha bisogno di modifiche in quanto è stata molto apprezzata dai ragazzi per la sua struttura. L'attualità dell'argomento ed il fatto di non introdurlo subito in modo formale ma attraverso un'analisi ed un dialogo di gruppo basati su quello che gli studenti già conoscevano dell'argomento si sono rivelati un'ottima unione. In particolare, l'approccio non rigido che gli studenti hanno avuto con l'argomento ha permesso a tutti loro di potersi esprimere senza timore e così esplicitare le loro idee e dubbi. Anche le due prove intermedie si sono rivelate fondamentali e credo che lo siano state ancor più per i ragazzi che per me. Senza quelle prove i ragazzi avrebbero solo

visto un gran numero i concetti e nulla sarebbe rimasto loro in mente, invece lavorando attivamente con questi concetti attraverso esercizi hanno potuto entrare a fondo nell'argomento e le lezioni non sono state solamente una mia spiegazione ma, prima di tutto, un loro lavoro. Per quanto riguarda la parte teorica, non le apporterei sostanziali modifiche in quanto gli obiettivi 2) e 3) sono stati raggiunti da entrambe le classi e gli obiettivi 1) e 4) da almeno una delle due classi. Come ho detto, semplificherei l'obiettivo 4) eliminando la parte relativa alle matrici irriducibili, in quanto il procedimento per ottenerle è molto pesante dal punto di vista del calcolo, e mi concentrerei maggiormente sulla definizione di Pagerank per evitare che possano nascere misconcezioni fra il concetto di Pagerank e l'algoritmo Google's Pagerank.
Per quanto riguarda la parte pratica, delle modifiche sono necessarie. Innanzi tutto, anche se nel questionario la maggioranza degli studenti ha scritto di ritenere adeguato il numero di ore dedicate a Matlab, credo che sia necessario aumentarle per due principali motivi: per fare in modo che gli studenti abbiamo una maggiore padronanza del software e per poter analizzare con più calma l'algoritmo Google's Pagerank. Credo che quest'ultimo aspetto sia il più importante in quanto è il coronamento del progetto ma, purtroppo, solo pochissimi ragazzi di entrambe le classi hanno capito il funzionamento di tale algoritmo.
In conclusione mi ritengo soddisfatta del progetto e dei suoi risultati, i quali ci hanno permesso di capire quanto i nostri studenti abbiamo bisogno, al giorno d'oggi, di confrontarsi con argomenti di Matematica attuali e che permettano loro di avvicinarsi in modo serio ma non troppo rigido alla programmazione. Con le opportune modifiche, di cui abbiamo parlato in questo capitolo, questo progetto può essere riproposto in una qualsiasi classe di terza, quarta o quinta superiore di un Liceo Scientifico permettendo ai ragazzi di conoscere e lavorare con argomenti di ricerca attuale.

Appendice

Funzione *pagerankpow* che sfrutta il metodo delle potenze:

```
function [x,iter] = pagerankpow(G)
%Conventional power method
G = G - diag(diag(G));
[n,n] = size(G);
p = 0.85;
delta = (1-p)/n;
c = sum(G,1);
k = find(c~=0);
D = sparse(k,k,1./c(k),n,n);
e = ones(n,1);
z = ((1-p)*(c~=0) + (c==0))/n;
A = p*G*D + e*z;
x = e/n;
oldx = zeros(n,1);
iter=0;
while norm(x - oldx) > .01
oldx = x;
x = A*x;
iter=iter+1;
end
x = x/sum(x);
% Bar graph
```

```
bar(x)
title('Page Rank - Potenze')
```

Funzione *pagerank* che risolve un sistema lineare:

```
function x = pagerank(U,G,p)
% Pagerank  Google's Pagerank
if nargin < 3, p = .85; end
G = G - diag(diag(G));
[n,n] = size(G);
c = sum(G,1);
r = sum(G,2);
k = find(c~=0);
D = sparse(k,k,1./c(k),n,n);
% Solve (I - p*G*D)*x = e
e = ones(n,1);
I = speye(n,n);
x = (I - p*G*D)\e;
x = x/sum(x);
% Bar graph of page rank.
shg
bar(x)
title('Page Rank')
```

Bibliografia

[1] Austin, D. (2007). How Google finds your needle in the web's haystack. *Grand valley state university.*

[2] Berry, M.W., Drmac, Z., Jessup, E.R. (1998). Matrices, vector spaces and information retrieval. *SIAM Review.*

[3] D'Amore, B. (1999). *Elementi di Didattica della Matematica.* Pitagora Editrice Bologna.

[4] Elden, L. (2005). *Matrix methods in data mining and pattern recognition.* SIAM.

[5] Farahat, A., Lofaro T., Miller, J.C., Rae, G., Ward, L.,A. (2004). Authority rankings from Hits, Pagerank and Salsa: existence, uniquess and effect of initilization. *SIAM Journal on Scientific Computing.*

[6] Langville, A.N.,Meyer, C.D. (2005). A survey of eigenvector methods for web information retrieval. *SIAM Review.*

[7] Moler, C. (2011). *Experiments with Matlab.* www.mathworks.com.

[8] Moler, C. (2004). *Numerical computing with Matlab.* SIAM.