Laboratorio didattico di matematica computazionale

Beatrice Meini

Lezione 8 - 30/4/2014

1 Matrici sparse

Le matrici sparse sono matrici che hanno un numero basso di elementi non nulli, rispetto alla dimensione della matrice stessa. Matrici sparse sono ad esempio le matrici tridiagonali, oppure matrici a banda, a "freccia", oppure matrici che non hanno un particolare pattern, ma che hanno pochi elementi diversi da zero.

Octave permette di gestire le matrici sparse memorizzando solo gli elementi diversi da zero e i corrispondenti indici di riga e di colonna. La matrice identità è una matrice sparsa, che puè essere definita tale usando il comando `speye`:

```
octave:2> n = 10;
octave:3> A = speye(n)
A =

Compressed Column Sparse (rows = 10, cols = 10, nnz = 10)

  (1, 1) ->  1
  (2, 2) ->  1
  (3, 3) ->  1
  (4, 4) ->  1
  (5, 5) ->  1
  (6, 6) ->  1
  (7, 7) ->  1
  (8, 8) ->  1
  (9, 9) ->  1
  (10, 10) ->  1
```

Le matrici sparse sono rappresentare nnz da terne, dove nnz è il numero di elementi diversi da zero, e ciascuna terna è costituita da indice di riga, indice di colonna, e elemento corrispondente. In pratica, vengono memorizzati solo gli elementi diversi da zero.

La sparsità può essere visualizzata con il comando `spy`:

```
octave:3> spy(A)
```

Una generica matrice può essere convertita in una matrice sparsa mediante il comando `sparse`. Ad esempio:

```
octave:34> A = [12  3  0  0  0; -1  0  0  0  1;   7  0  2  0  0]
A =
```

```
   12     3     0     0     0
   -1     0     0     0     1
    7     0     2     0     0

octave:35> A = sparse(A)
A =

Compressed Column Sparse (rows = 3, cols = 5, nnz = 6)

  (1, 1) ->  12
  (2, 1) -> -1
  (3, 1) ->  7
  (1, 2) ->  3
  (3, 3) ->  2
  (2, 5) ->  1
```

In generale è comunque preferibile definire direttamente una matrice sparsa, piuttosto che trasformarla in sparsa mediante il comando `sparse`. Per creare matrici sparse si possono utilizzare comandi specifici, descritti ad esempio qui: `http://www.gnu.org/software/octave/doc/interpreter/Creating-Sparse-Matrices.html`

Il prodotto matrice/vettore nel caso in cui la matrice è sparsa viene fatto moltiplicando solo gli elementi diversi da zero della matrice. Per verificare sperimentalmente che il prodotto matrice/vettore con matrici sparse è più veloce, creiamo la matrice sparsa

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \tag{1}$$

di dimensione $n = 10000$, la moltiplichiamo per un vettore random e confrontiamo il tempo di calcolo con quello che otterrei con la matrice non sparsa (si veda l'help per il funzionamento di `cputime`, `sparse` e `full`):

```
octave:20> n=10000;
octave:21> A=2*speye(n)-sparse([1:n-1],[2:n],ones(n-1,1),n,n)-
sparse([2:n],[1:n-1],ones(n-1,1),n,n);
octave:22> x=rand(n,1);
octave:23> t=cputime(); y=A*x; t=cputime()-t
t =  4.00000000000134e-03
octave:24> FA=full(A);
octave:25> t=cputime(); y=FA*x; t=cputime()-t
t =  1.40008000000002e-01
```

Attenzione: I tempi naturalmente dipendono dalla macchina. Se non apprezzate la differenza tra aritmetica sparsa e non, aumentate il valore di n.

Dare il comando "format short e" per visualizzare meglio i tempi.

Esercizio 1. Si costruisca una function `tf=tempomoltf(p)` che prende in input un intero $p > 1$ e restituisce in output il vettore `tf` tale che `tf(i)` è il tempo di CPU utilizzato per effettuare il prodotto Ax con A di dimensione 2^{6+i}, $i = 1, \ldots, p$, dove A è definita in (1), nel caso in cui A sia "full".

Quale è il valore massimo di p che potete usare senza avere un messaggio di errore del tipo "memory exausted"?

Prendere tale p e tracciare il grafico in scala semilogaritmica di `tf`.

Esercizio 2. Si costruisca una function `ts=tempomolts(p)` che prende in input un intero $p > 1$ e restituisce in output il vettore `ts` tale che `ts(i)` è il tempo di CPU utilizzato per effettuare il prodotto Ax con A di dimensione 2^{6+i}, $i = 1, \ldots, p$, dove A è definita in (1), nel caso in cui A sia "sparse".

Quale è il valore massimo di p che potete usare senza avere un messaggio di errore del tipo "memory exausted"?

Prendere tale p e tracciare il grafico in scala semilogaritmica di `ts`.

Anche certe operazioni su matrici sono specializzate a matrici sparse. Ad esempio i comandi `lu`, `chol`, /, se applicati a matrici sparse cercano di sfruttarne la sparsità.

Si contruisca una matrice sparsa $n \times n$ che ammette fattorizzazione LU in questo modo:

```
octave:15> n=100;
octave:16> I=randperm(n);
octave:17> J=randperm(n);
octave:18> A=sparse(I,J,rand(n,1),n,n)+speye(n);
```

Si calcoli la fattorizzazione LU:

```
octave:19> [L,U]=lu(A);
```

Osservare che L e U sono matrici "sparse".

Esercizio 3. Si costruisca una function `tluf=tempofattf(p)` che prende in input un intero $p > 6$ e restituisce in output il vettore `tluf` tale che `tluf(i)` è il tempo di CPU utilizzato per calcolare la fattorizzazione LU nel caso in cui A sia "full", dove A è costruita come nelle istruzioni sopra e ha dimensione 2^{6+i}, $i = 1, \ldots, p$.

Dare il comando "format short e" per visualizzare meglio gli elementi del vettore "tluf".

Quale è il valore massimo di p che potete usare senza avere un messaggio di errore del tipo "memory exausted"?

Prendere tale p e tracciare il grafico in scala semilogaritmica di `tluf`.

Esercizio 4. Si costruisca una function `tlus=tempofatts(p)` che prende in input un intero $p > 6$ e restituisce in output il vettore `tlus` tale che `tlus(i)` è il tempo di CPU utilizzato per calcolare la fattorizzazione LU nel caso in cui A sia "sparse", dove A è costruita come nelle istruzioni sopra e ha dimensione 2^{6+i}, $i = 1, \ldots, p$.

Dare il comando "format short e" per visualizzare meglio gli elementi del vettore "tlus".

Quale è il valore massimo di p che potete usare senza avere un messaggio di errore del tipo "memory exausted"?

Prendere tale p e tracciare il grafico in scala semilogaritmica di `tlus`.

2 Il "Pagerank" di Google

Il successo del motore di ricerca Google è dovuto alla capacità di ordinare, in ordine di importanza decrescente, le pagine web che rispondono ad una ricerca

(query) dell'utente. Infatti, generalmente, solo le prime pagine web elencate sono quelle che interessano all'utente.

L'ordinamento delle pagine web si basa su un modello matematico di navigazione sul web. Secondo questo modello, un utente che naviga nel web "di norma" sceglie con distribuzione di probabilità uniforme un link dalla pagina che sta visitando; invece talvolta l'utente decide di scegliere, ancora con distribuzione di probabilità uniforme, una pagina web scelta tra tutte le pagine web esistenti. Secondo il modello di Google, la pagina web w_1 è *più importante* della pagina web w_2 se, facendo tendere il tempo di navigazione all'infinito, l'utente visita la pagina w_1 con probabilità maggiore rispetto alla pagina w_2.

Questo concetto può essere formalizzato matematicamente nel seguente modo. Sia W l'insieme delle pagine web che possono essere raggiunte seguendo link successivi a partire da qualche pagina fissata, e sia n la cardinalità di W. L'insieme W varia nel tempo; l'insieme utilizzato da Google è formato da circa $4 \cdot 10^9$ pagine! Sia $G = (g_{i,j})$ la matrice $n \times n$ che rappresenta le connessioni tra una pagina e l'altra: $g_{i,j} = 1$ se c'è un link dalla pagina j alla pagina i, e $g_{i,j} = 0$ altrimenti. La matrice G può avere enorme dimensione, ma è molto sparsa.

Sia c_j la somma degli elementi sulla j-esima colonna di G:

$$c_j = \sum_i g_{i,j}$$

Sia p la probablità che l'utente segua un link dalla pagina che sta visitando (dunque $1 - p$ è la probablità che l'utente scelga una pagina arbitraria). Definiamo la matrice $A = (a_{i,j})$ di dimensione $n \times n$:

$$a_{i,j} = \begin{cases} p\frac{g_{i,j}}{c_j} + \frac{1-p}{n}, & se \;\; c_j \neq 0 \\ \frac{1}{n}, & se \;\; c_j = 0 \end{cases}$$

Si osservi che per costruire la matrice A si scala la matrice G mediante la somma degli elementi su ciascuna colonna. Se la somma degli elementi lungo la colonna j è nulla, significa che non ci sono link che partono dalla pagina web j; in questo caso assegnamo una probabilità uniforme, uguale a $1/n$, a tutti gli elementi della j-esima colonna.

Valori tipici usati nel calcolo del Pagerank di Google sono $n = 4 \cdot 10^9$ and $p = 0.85$; per questi valori $(1 - p)/n = 3.75 \cdot 10^{-11}$.

I comandi

```
octave:2> n = 10;
octave:3> G = sprand(n,n,0.1);
```

generano una matrice 10×10 random G sparsa, dove 0.1 è la densità di sparsità (si veda l'help per il funzionamento). Per trasfomare G in una matrice con elementi uguali a 0 o 1 possiamo dare il comando

```
octave:4> G=(G!=0)
G =

Compressed Column Sparse (rows = 10, cols = 10, nnz = 10)

  (1, 1) ->  1
  (9, 2) ->  1
```

```
  (1, 4) ->  1
  (10, 4) ->  1
  (10, 7) ->  1
  (6, 8) ->  1
  (7, 8) ->  1
  (10, 8) ->  1
  (10, 9) ->  1
  (5, 10) ->  1
```

Per visualizzare tutti gli elementi della matrice G occorre diamo comando

```
octave:7> full(G)
ans =

   1   0   0   1   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   1
   0   0   0   0   0   0   0   1   0   0
   0   0   0   0   0   0   0   1   0   0
   0   0   0   0   0   0   0   0   0   0
   0   1   0   0   0   0   0   0   0   0
   0   0   0   1   0   0   1   1   1   0
```

Il comando

```
octave:8> spy(G)
```

mostra graficamente gli elementi diversi da zero.

Esercizio 5. Si costruisca una matrice G sparsa di dimensione 100, con elementi 0 o 1, e si costrisca la matrice A, scegliendo $p = 0.85$. Si verifichi che la matrice A ha elementi non negativi, ed ha somma degli elementi uguali a 1 su ogni colonna. Per calcolare la somma degli elementi su ogni colonna si utilizzi l'istruzione `sum`.

È possibile dimostrare che esiste un unico vettore x, con elementi non negativi, tale che

$$x = Ax, \quad \sum_{i=1}^{n} x_i = 1.$$

Il vettore x rappresenta il Pagerank di Google: la pagina corrispondente all'indice j è più importante della pagina corrispondente a i se $x_j > x_i$. Infatti, x_j è la probabilità che, per il tempo che tende a infinito, l'utente visiti la pagina j.

Un metodo per calcolare il vettore x è generare la successione di vettori:

$$x^{(0)} = [1/n, \ldots, 1/n]^T, \quad x^{(k+1)} = Ax^{(k)} \quad k \geq 0. \tag{2}$$

Infatti la successione $\{x^{(k)}\}$ converge a x (se volete saperne di più cercate con Google "power method").

Esercizio 6. Definire la `function x=pow(A,maxit,tol)` che prende in input la matrice A, un numero massimo di iterazioni `maxit` e una tolleranza `tol`, e che restituisce in output una approssimazione del vettore x ottenuta mediante l'iterazione (2), dove il calcolo della successione è interrotto se $k =$`maxit` oppure $||x^{(k)} - x^{(k-1)}||_1 <$`tol`.

Il grosso difetto della function `pow` è che non sfrutta la sparsità della matrice G. La successione $x^{(k)}$ infatti può essere calcolata non calcolando esplicitamente la matrice A. Infatti la matrice A può essere scritta come

$$A = pGD + ez^T$$

dove e è il vettore con tutte le componenti uguali a 1, D è la matrice diagonale con elementi diagonali

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

e z è il vettore con componenti

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

Dunque si ottiene che

$$x^{(k+1)} = pGDx^{(k)} + e(z^T x^{(k)}). \tag{3}$$

In questo modo per il calcolo di $x^{(k+1)}$ viene sfruttata la sparsità di G.

Le istruzioni

```
octave:8> c=sum(G,1);
octave:9> k=find(c!=0);
octave:10> D = sparse(k,k,1./c(k),n,n);
octave:11> e = ones(n,1);
octave:12> p=0.85;
octave:13> Gs=p*G*D;
octave:14> z = ((1-p)*(c!=0) + (c==0))/n;
```

costruiscono la matrice D sparsa, la matrice $Gs = pGD$ sparsa e il vettore z.

Esercizio 7. Definire la `function x=pow2(G,maxit,tol)`, modificando la `pow`, in modo che la successione di vettori sia calcolata mediante la formula (3). Scegliere n abbastanza grande, e confrontare i risultati di `pow` e `pow2`.

2.1 Un esempio di piccola rete

Salvare il file `www.dm.unipi.it/~meini/LDMC14/harvard500.mat` e dare il comando `load harvard500.mat`. La matrice G è la matrice dei link di una sottorete 500×500 ottenuta partendo dal sito `www.harvard.edu`; il vettore U è un array di stringhe, che contiene i nomi delle pagine web.

Dare il comando `spy(G)` per vedere la struttura di G.

Esercizio 8. Calcolare il vettore x mediante `pow2`. Successivamente visualizzare il Pagerank e le pagine corrispondenti mediante i comandi:

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

% Print URLs in page rank order.

   [ignore,q] = sort(-x);
   disp('      page-rank   url')
```

```
k = 1;
while (k <= n) & (x(q(k)) >= .005)
    j = q(k);
    disp(sprintf(' %3.0f %8.4f  %s', j,x(j),U{j}))
    k = k+1;
end
```

3 Esercizi da inviare al docente

Inviare per e-mail, con subject "LDMC: lezione 8, [cognome, nome]":

1. Le function scritte per svolgere gli esercizi 3 e 4, e i vettori dei tempi ottenuti nei due esercizi con i rispettivi valori massimi consentiti di p per non avere "memory exausted".

2. La function scritta per svolgere l'esercizio 7 e le prime 5 componenti del vettore ottenuto prendendo come matrice G quella ottenuta dal file `harvard500.mat`.