Laboratorio Sperimentale di Matematica Computazionale (Parte I) Lezione 2

Gianna Del Corso <gianna.delcorso@unipi.it>

3 Marzo 2017

Laboratorio Sperimentale di Matematica Computazionale (Parte I) Lezione 2

Gianna Del Corso <gianna.delcorso@unipi.it>

3 Marzo 2017

1 Comandi utili per lavorare con matrici sparse

Le matrici sparse sono una classe speciale di matrici con la caratteristica di contenere un significativo numero di elementi nulli. Con `Matlab` è possibile manipolare matrici sparse, memorizzandole in forma compatta, ossia memorizzando solo gli elementi non nulli insieme ai loro indici. In questo modo si può notevolmente ridurre lo spazio per la memorizzazione della matrice.

Consideriamo una matrice $m \times n$ con $n$nz elementi non nulli. `Matlab` utilizza tre array per memorizzare internamente matrici sparse a elementi reali, e occorrono solo $n$nz locazioni di memoria per memorizzare l'intera matrice, indipendentemente dalla sua dimensione.

La densità di una matrice è il rapporto tra il numero di elementi non nulli e il numero totale di elementi della matrice. Se tale valore è basso è quindi conveniente usare la memorizzazione sparsa.

Ecco i principali comandi per lavorare con matrici sparse: si rimanda all'help di linea per una completa descrizione delle varie possibilità di uso.

- Conversione di matrici da dense a sparse

  `B=sparse(A)` dove A è densa e B risulterà sparsa

  `A=full(B)` fa l'operazione contraria

- Creazione diretta di matrici sparse

  `S=sparse(I, J, s, m, n, nzmax)` dove I e J sono vettori che contengono indici di riga e colonna degli elementi non nulli di S. Il vettore s contiene i valori dei nonzero: $S(I(k), J(k)) = s(k)$. Le dimensioni della matrice sono specificate da m (numero di righe) ed n (numero di colonne) e $nzmax$ è il numero massimo di elementi non nulli di S.

`S=spdiags(B, d, m, n)` crea una matrice sparsa $m \times n$, ove le colonne di B sono le diagonali della matrice S e il vettore **d**, avente tante componenti quante sono le colonne di B, contiene l'informazione sul numero di diagonale contenuta nella corrispondente colonna di B.

```
>> v=ones(5, 1);
>> S=spdiags([-v, 2*v, -v], -1:1, 5, 5)
S =

   (1,1) 2
   (2,1) -1
   (1,2) -1
   (2,2) 2
   (3,2) -1
   (2,3) -1
   (3,3) 2
   (4,3) -1
   (3,4) -1
   (4,4) 2
   (5,4) -1
   (4,5) -1
   (5,5) 2
>> full(S)
ans =

    2 -1 0 0 0
   -1 2 -1 0 0
    0 -1 2 -1 0
    0 0 -1 2 -1
    0 0 0 -1 2
```

`I=speye(n)` genera la versione sparsa della matrice identità, è l'equivalente sparso di `eye(n)`.

Il comando `B=sprand(A)` o `B=sprand(m,n, d)`, genera una matrice sparsa con elementi generati a caso con una distribuzione uniforme tra 0 e 1. Se è specificata una matrice A sparsa, la struttura della matrice generata a caso è uguale a quella di A. In caso contrario, è generata una matrice $m \times n$ con la densità d specificata.

`spones(A)` fornisce una matrice con la stessa struttura di A ma con elementi non nulli rimpiazzati da 1.

- Per importare una matrice sparsa dentro `Matlab`, si può creare un file che contiene in ogni riga tre informazioni: indice di riga, di colonna e valore di ciascun elemento non nullo. Si carica tale file di formato `.dat` in `Matlab` con il comando `load('nome.dat')` e poi si usa `spconvert` per convertire i dati letti in una matrice sparsa.

```
>> B=load('A.mat')
B =

    1 1 3
    1 2 7
    1 5 -2
    2 3 -1
    2 4 3
    5 1 -6
>>full(spconvert(B))

ans =

    3 7 0 0 -2
    0 0 -1 3 0
    0 0 0 0 0
    0 0 0 0 0
   -6 0 0 0 0
   >>
```

- Il comando `spy(A)` mostra in un plot la struttura di sparsità della matrice sparsa A mettendo un puntino per ogni elemento nonzero. Attenzione se A non è definita come matrice sparsa, ci saranno dei punti anche in corrispondenza di elementi di modulo molto piccolo.
- Il comando `nnz(a)` restituisce il numero di elementi non nulli di A.
- Il comando `[rows, cols, vals]=find(A)` trova gli indici di riga e colonna di A di valore vals.
- Molte funzioni Matlab permettono di lavorare su matrici sparse. Attenzione che somma o prodotto tra una matrice di tipo *sparse* e di tipo *double* fornisce una matrice di tipo *double* anche se la matrice ottenuta ha un numero di nonzeri tale da poter essere considerata sparsa. Se ne tenga conto quando si svolgono gli esercizi in modo da lavorare sempre con matrici di tipo *sparse*.

2 Grafi (in a nutshell)

Il grafo è una coppia ordinata $\mathcal{G} = (V, E)$ di insiemi dove V è l'insieme dei nodi ed $E, E \subseteq V \times V$ è l'insieme degli archi. Un grafo si dice orientato se gli archi hanno una direzione, altrimenti si dice non orientato.

Un cammino tra due nodi u_0 e u_m del grafo è una sequenza di nodi $u_0, u_1, \ldots, u_m$ tali che gli archi $(u_{i-1}, u_i) \in E$, per $i = 1, 2, \ldots, m$. Due nodi $u, v \in V$ sono *connessi* se esiste un cammino da u a v. Il numero di archi incidenti in un vertice $v \in V$ si chiama *grado* di v, e si indica con $deg(v)$. Per i grafi orientati si parla di *indegree* e *outdegree*.

Una *componente connessa* di un grafo non orientato $\mathcal{G}$ è un sottografo $\mathcal{G}' = (V', E')$ tale che ogni coppia di nodi $u, v \in V'$ risulta connessa e nessun altro nodo di $\mathcal{G}$ risulta connesso a nodi di $\mathcal{G}'$.

La *Matrice di Adiacenza* A di un grafo è una matrice di dimensione $|V| \times |V|$ tale che

$$A_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Se il grafo è non orientato la matrice di adiacenza è simmetrica, in quanto si considera come se il grafo avesse sia archi (i, j) che archi (j, i).

La *Matrice di incidenza* M è una matrice di dimensione $|V| \times |E|$ tale che la colonna di M corrispondente ad un arco (i, j) ha solo due nonzeri, uno nella riga i e uno nella riga j, uno con valore 1 e l'altro con valore -1. Se $\mathcal{G}$ è non orientato l'arco (i, j) non viene però ripetuto due volte.

Esercizio 1. Data una matrice di adiacenza A di un grafo *non orientato e connesso*. Sia L la matrice associata al grafo e definita come $L = D - A$ dove D è una matrice diagonale tale che $d_{ii} = \deg(v_i)$, cioè il grado del nodo v_i. Tale matrice è detta *Laplaciano* del grafo.

Scrivere una funzione

```
function L=graph_laplacian(A)
% A e' la matrice di adiacenza di un grafo non orientato
% L e' la matrice Laplaciana del grafo descritto da A
```

Attenzione a lavorare sempre con matrici sparse, in particolare utilizzare i comando per le mtrici sparse p costruire la matrice diagonale D.

Si faccia l'upload del grafo "Bucky" eseguendo i seguenti comandi [1]

```
>> [A, v]=bucky;
>> gplot(A, v);
```

Sul grafo "Bucky" di cui A è matrice di adiacenza:

(a) Si verifichi che L ha un autovalore nullo e che un corrispondente autovettore è il vettore $(1, 1, \ldots, 1)^T$. Si dimostri che questa proprietà vale per ogni grafo $\mathcal{G}$ non orientato.

(b) Si verifichi che L è semidefinita positiva e quindi ha autovalori non-negativi. Si osservi che questa è una proprietà generale della matrice Laplaciana.

(c) Utilizzando il comando `eigs` con gli opportuni parametri, si calcoli la *connettività algebrica* di $\mathcal{G}$ cioè il secondo autovalore (più piccolo) di L.

Esercizio 2. Sia $\mathbf{x}_2$ un autovettore associato all'autovalore λ_2. Questo autovettore si chiama *vettore di Fiedler* ed ha importanti proprietà collegate al concetto di comunità. Si scriva una funzione

[1] Per gli uteni Octave scaricare il grafo "Bucky" dalla piattaforma di e-learning eseguendo i comandi ivi descritti.

```
function [l,x,perm]=Fiedler_vector(A)
% A matrice di Adiacenza di un grafo
% x e' il vettore di Fiedler
% l e' il secondo autovalore piu' piccolo di L
% perm e' la permutazione indotta dall'ordinamento di x per valori crescenti
```

La funzione deve restituire: il valore `l` della connettività , il vettore `x` di Fiedler del grafo e il vettore di permutazione `perm` che riordina le sue componenti (cioè tale che $\mathbf{x}(\mathbf{perm})$ è un vettore le cui componenti sono ordinate in ordine crescente). Con il comando `spy` si visualizzi la struttura della matrice permutata in accordo con la permutazione `perm` e si verifichi che la sua ampiezza di banda è inferiore a quella della matrice A. Questo procedimento suggerisce una possibile euristica per la riduzione di banda di una matrice sparsa.

Esercizio 3. (a) Si scriva una funzione

```
function M=incidence_matrix(A)
% A matrice di adiacenza di un grafo
% M sua matrice di incidenza
```

facendo attenzione a lavorare solo sui valori nonzero nella parte triangolare superiore o inferiore per non duplicare gli archi nel caso di grafi non orientati.

(b) Si verifichi sull'esempio del grafo "Bucky" che, se il grafo è non orientato, la matrice Laplaciana L del grafo è tale che $L = MM^T$. Si noti inoltre che il $\text{rank}(M) = n - t$ dove t è il numero di componenti connesse del grafo, quindi nel nostro esempio"bucky" $\text{rank}(M) = n - 1$.

Esercizio 4. Per un grafo non orientato si costruisca la matrice aumentata

$$B = \begin{pmatrix} I & M^T \\ M & 0 \end{pmatrix},$$

dove M è la matrice di incidenza.

Per un dato vettore $\mathbf{b}$, si assuma che il sistema singolare $L\mathbf{x} = \mathbf{b}$ abbia soluzione [2] $\mathbf{x}$. Si dica come deve essere fatto il termine noto $\mathbf{c}$ affinchè la soluzione $\mathbf{y}$ del sistema aumentato $B\mathbf{y} = \mathbf{c}$ contenga al suo interno la soluzione del sistema $L\mathbf{x} = \mathbf{b}$. Si scriva una funzione `x=solve_augmented(M, b)` che permetta di ottenere la soluzione del sistema $L\mathbf{x} = \mathbf{b}$ dalla soluzione del sistema $B\mathbf{y} = \mathbf{c}$.

Per risolvere il sistema $B\mathbf{y} = \mathbf{c}$ si suggerisce di utilizzare la fattorizzazione QR con pivot sulle colonne; il fattore triangolare R che otteniamo avrà le ultime t righe nulle se $\dim(N(B)) = t$. Si ricordi di permutare il vettore soluzione perchè i fattori Q ed R sono tali che $BP = QR$, con P matrice di permutazione restituita dalla funzione Matlab `[Q, R, P]=qr(B)`, quindi $(BP)(P^T\mathbf{y}) = \mathbf{c}$.

In particolare si può completare la seguente funzione

[2] Poichè L è singolare, si costruisca $\mathbf{b} \in \text{R}(L)$.

```
function [ x ] = solve_augmented( M, b )
% USO: x=solve_augmented(M, b),
% M matrice di incidenza di un grafo,
% b vettore dei termini noti del sistema lineare Lx=b

[n, m]=size(M);
B=...; %B deve essere sparsa
[Q, R, P]=qr(...); %utilizziamo il metodo qr con pivot sulle
% colonne per calcolare una fattorizzazione
% "rank-revealing" cioe' dove le ultime t righe di R sono nulle
% se rank(R)=n-t;

c1=rand(m, 1);
c2=....;
c=[c1;c2];
z=...; %soluzione del sistema Qz=c
r=.... ; % rango di R.. attenzione si opera in aritmetica finita...

if abs(z(r+1:n+m))>10^-12*norm(z) || r==n+m
%controlla che z(r+1:n+m)==0 e che R sia singolare
   error('qualcosa va male, vettore b non adeguato? ');
end
R1=....; %minore principale di testa di R di rango pieno
z1=z(1:r);
V1=R(1:r, ...); %prime r righe delle ultime colonne della matrice R
alpha=... ; % vettore qualsiasi di dimensioni opportune...
w1=R1\(z1-V1*alpha);
w=[w1;alpha];
y=...; % occorre permutare le componenti di w
x=y(...);

% verifica che x sia soluzione di Lx=b

end
```