Sistemi lineari sovradeterminati e SVD

4 luglio 2007

1 Sistemi lineari sovradeterminati

Si consideri il problema

$$\begin{aligned} x_1 + x_2 &= 1 \\ x_1 - x_2 &= 0 \\ x_1 + 3x_2 &= 0 \end{aligned}$$

Risulta chiaro che il sistema non ha soluzione. Infatti l'unica soluzione delle prime due equazioni è $x_1 = x_2 = 1/2$, che però non verifica $x_1 + 3x_2 = 0$.

Ritrascrivendo tutto in termini matriciali, possiamo dunque affermare che il problema $Ax = b$, con

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

e $b = (1,0,0)$ non ha soluzione.

In alternativa, risulta quindi ragionevole calcolare $x^* = (x_1^*, x_2^*)$ tale che

$$\gamma := \min_{x \in \mathscr{C}^n} \|b - Ax\|_2 = \|b - Ax^*\|_2 \tag{1}$$

dove

$$\|u\|_2 = \sqrt{\sum_i u_i^2}.$$

La prima questione riguarda l'esistenza e unicità di tale minimo x^*. A tal proposito citiamo il segue teorema (cf. [1], p. 432)

Teorema 1.1 *Sia X l'insieme dei vettori di $\mathscr{C}^n$ tali che $\hat{x} \in X$ se e solo se*

$$\min_{x \in \mathscr{C}^n} \|b - Ax\|_2 = \|b - A\hat{x}\|_2 \tag{2}$$

Supponiamo $A \in \mathscr{C}^{m \times n}$ con $m \geq n$ e $b \in \mathscr{C}^n$. Valgono le seguenti proprietà

1. *$x \in X$ se e solo se*

$$A^H Ax = A^H b, \tag{3}$$

cioé x risolve il sistema delle equazioni normali.

2. *X è un insieme non vuoto, chiuso e convesso.*

3. *l'insieme X si riduce ad un solo elemento x^* se e solo se la matrice A ha rango massimo.*

4. *Esiste $x^* \in X$ tale che*
$$\|x^*\|_2 = \min_{x \in X} \|x\|_2.$$
Tale x^* è detto soluzione di minima norma.

In altre parole, se A ha rango n allora X ha un unico elemento, mentre se A ha rango minore di n allora X ha un unico elemento di minima norma 2.

2 Alcuni metodi per risolvere sistemi sovradeterminati.

2.1 Risoluzione equazioni normali

Supponiamo A abbia rango n. Dal teorema 1.1 sappiamo che la soluzione cercata risolve $A^H Ax = A^H b$. Nel caso A abbia coefficienti in $\mathscr{R}$, ció si riduce a risolvere $A^T Ax = A^T b$. Se $LL^H = A^H A$ per L triangolare inferiore (fattorizzazione di Cholesky) basta risolvere

$$Ly = A^H b,$$
$$L^H x = y.$$

Il costo computazionale è di $n^2 m/2$ operazioni moltiplicative per la costruzione di $A^H A$ e di $n^3/6$ moltiplicative per la soluzione del sistema, e quindi il costo totale è di
$$n^2 m/2 + n^3/6$$
operazioni moltiplicative. Osserviamo che se A non ha rango massimo non si puó applicare il metodo di Cholesky ma il metodo di Gauss con variante di massimo pivot.

2.2 Metodo QR

Data una matrice $A \in \mathscr{C}^{m \times n}$ esistono $Q \in \mathscr{C}^{m \times n}$ unitaria (cioè $QQ^H = I_m$, $Q^H Q = I_n$) ed $R \in \mathscr{C}^{n \times n}$ triangolare superiore tali che $A = QR$. Si osservi che a seconda degli autori la fattorizzazione QR sopraindicata viene sostituita con la fattorizzazione $A = QR$ con $Q \in \mathscr{C}^{m \times m}$ unitaria ed $R \in \mathscr{C}^{m \times n}$ triangolare superiore cioè

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

con $R_1 \in \mathscr{C}^{n \times n}$ triangolare superiore.

La toolbox di Matlab

```
[q,r]= qr(A)
```

esegue la prima fattorizzazione. Conseguentemente considereremo tale caso, con $Q \in \mathscr{C}^{m\times n}$ unitaria e $R \in \mathscr{C}^{n\times n}$ triangolare superiore. Nota la fattorizzazione qr si deduce che $Rx = Q^H QRx = Q^H b \in \mathscr{R}^n$.

Se A ha *rango massimo* allora R è non singolare e quindi il problema $Rx = Q^H b$ risolto facilmente per sostituzione all.indietro. Il costo computazionale per la fattorizzazione QR è di $n^2(m-n/3)$, il costo computazionale della risoluzione del sistema triangolare è $n^2/2$ operazioni moltiplicative. Quindi il costo totale è

$$n^2(m-n/3)+n^2/2.$$

Se A non ha *rango massimo* allora $AE = QR$ con E matrice di permutazione,

$$R = \begin{pmatrix} R_1 & S \\ 0 & 0 \end{pmatrix}$$

dove $R \in \mathscr{C}^{n\times n}$, $R_1 \in \mathscr{C}^{r\times r}$ sono matrici triangolari superiori. In merito l'help di Matlab fornisce le seguenti indicazioni:

```
>> help qr

 QR     Orthogonal-triangular decomposition.
    [Q,R] = QR(A) produces an upper triangular matrix R of the same
    dimension as A and a unitary matrix Q so that A = Q*R.

    [Q,R,E] = QR(A) produces a permutation matrix E, an upper
    triangular R and a unitary Q so that A*E = Q*R.  The column
    permutation E is chosen so that abs(diag(R)) is decreasing.

    [Q,R] = QR(A,0) produces the "economy size" decomposition.
    If A is m-by-n with m > n, then only the first n columns of Q
    are computed.

    [Q,R,E] = QR(A,0) produces an "economy size" decomposition in
    which E is a permutation vector, so that Q*R = A(:,E).  The column
    permutation E is chosen so that abs(diag(R)) is decreasing.

    By itself, QR(A) is the output of LAPACK'S DGEQRF or ZGEQRF routine.
    TRIU(QR(A)) is R.

    For sparse matrices, QR can compute a "Q-less QR decomposition",
    which has the following slightly different behavior.

    R = QR(A) returns only R.  Note that R = chol(A'*A).
    [Q,R] = QR(A) returns both Q and R, but Q is often nearly full.
    [C,R] = QR(A,B), where B has as many rows as A, returns C = Q'*B.
    R = QR(A,0) and [C,R] = QR(A,B,0) produce economy size results.

    The sparse version of QR does not do column permutations.
    The full version of QR does not return C.
```

```
    The least squares approximate solution to A*x = b can be found
    with the Q-less QR decomposition and one step of iterative refinement:

        x = R\(R'\(A'*b))
        r = b - A*x
        e = R\(R'\(A'*r))
        x = x + e;

    See also LU, NULL, ORTH, QRDELETE, QRINSERT, QRUPDATE.

>>
```

Nel caso della matrice

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

il comando `[Q,R,E] = qr(A)` fornisce

```
>> A=[1 1; 1 -1; 1 3]

A =

     1     1
     1    -1
     1     3

>> [Q,R,E]=qr(A)

Q =

   -0.3015   -0.4924   -0.8165
    0.3015   -0.8616    0.4082
   -0.9045   -0.1231    0.4082


R =

   -3.3166   -0.9045
         0   -1.4771
         0         0


E =

     0     1
     1     0
```

```
>> A*E-Q*R

ans =

  1.0e-015 *

    0.2220    0.1110
    0.2220    0.2220
   -0.4441         0

>>
```

Quindi dal punto di vista implementativi è facile calcolare la fattorizzazione QR di A anche quando il rango A non è massimo. Si stabilisce facilmente il rango r di A cercando il numero di elementi diagonali di R non nulli. Quindi la soluzione di norma minima risulta $x^* = (y_1 y_2)$, $y_1 \in \mathscr{C}^r$, $y_2 = (0,\dots,0) \in \mathscr{C}^{n-r}$ con $y_1 = R_1^{-1} c_1$, per $c_1 = ((Q^H b)_{ij})_{i,j=1,\dots,r}$.

2.3 Metodo SVD

Il termine SVD sta per *singular value decomposition*. Cominciamo col seguente teorema.

Teorema 2.1 *Sia $A \in \mathscr{C}^{m\times n}$. Allora esistono due matrici unitarie $U \in \mathscr{C}^{m\times m}$, $V \in \mathscr{C}^{n\times n}$ e una matrice diagonale $\Sigma \in \mathscr{C}^{m\times n}$ avente elementi σ_{ij} nulli per $i \neq j$ e uguali a σ_i per $i = j$ con*

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0 \tag{4}$$

tali che

$$A = U\Sigma V^H.$$

Siano u_i, v_i le i-sime righe rispettivamente di U e V. Per capire come risolvere un sistema lineare sovradeterminato tramite questa fattorizzazione ci affidiamo al seguente teorema

Teorema 2.2 *Sia $A \in \mathscr{C}^{m\times n}$ di rango r con $m \geq n \geq r$ e sia*

$$A = U\Sigma V^H$$

la decomposizione ai valori singolari di A. Allora la soluzione di minima norma è data da

$$x^* = \sum_{i=1}^{k} \frac{u_i^H b}{\sigma_i} v_i$$

e

$$\gamma^2 = \sum_{i=k+1}^{m} |u_i^H b|^2$$

dove al solito

$$\gamma := \min_{x\in\mathscr{C}^n} \|b - Ax\|_2 = \|b - Ax^*\|_2. \tag{5}$$

2.4 Alcune osservazioni su SVD

1. Osserviamo che Matlab dispone del comando svd. La descrizione dell'help è la seguente:

```
>> help svd

 SVD    Singular value decomposition.
    [U,S,V] = SVD(X) produces a diagonal matrix S, of the same
    dimension as X and with nonnegative diagonal elements in
    decreasing order, and unitary matrices U and V so that
    X = U*S*V'.

    S = SVD(X) returns a vector containing the singular values.

    [U,S,V] = SVD(X,0) produces the "economy size"
    decomposition. If X is m-by-n with m > n, then only the
    first n columns of U are computed and S is n-by-n.

    See also SVDS, GSVD.

 Overloaded methods
    help sym/svd.m

>>
```

Per fare pratica con questo comando sia

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

e utilizziamo `[U,S,V] = svd(A)` per avere la decomposizione SVD della matrice A. Il risultato è

```
>> A=[1 1; 1 -1; 1 3]

A =

     1     1
     1    -1
     1     3

>> [U,S,V] = svd(A)

U =

    0.3651    0.4472   -0.8165
```

```
   -0.1826    0.8944    0.4082
    0.9129   -0.0000    0.4082


S =

    3.4641         0
         0    1.4142
         0         0


V =

    0.3162    0.9487
    0.9487   -0.3162

>> X = U*S*V'

X =

    1.0000    1.0000
    1.0000   -1.0000
    1.0000    3.0000

>>
```

2. Si puó dimostrare che il rango della matrice è esattamente il numero di σ_i non nulli. In realtá la tentazione di determinarlo in virtú dei termini diagonali di Σ puó risultare fallace in quanto il computer fornisce solo un'approssimazione dei σ_i e qualora questi siano molto piccoli non si puó determinare con sicurezza il rango della matrice.

3 Facoltativo: Un esempio: compressione immagini.

Consideriamo i siti web:

```
1. http://amath.colorado.edu/courses/4720/2000Spr/Labs/SVD/svd.html
2. http://www.mathworks.com/company/newsletters/news_notes/
          oct06/clevescorner.html
3. http://www.mathworks.com/matlabcentral/fileexchange/
          loadFile.do?objectId=4772&objectType=File
```

L'argomento esposto è la compressione di immagini via SVD. In particolare si accenna ad una interessante implementazione in Matlab (non funziona in Octave!) relativa alla compressione di immagini. Entriamo nei dettagli. Sia $A = U\Sigma V^T$ la fattorizzazione SVD della matrice A. Se σ_i sono gli elementi diagonali della matrice diagonale e rettangolare Σ, u_k la k-sima colonna di U, v_k la k-sima colonna di V, allora

$$A = \sum_{k=1}^{n} \sigma_k u_k v_k^T.$$

Se l'indice k viene ordinato cosicchè i valori singolari siano in ordine decrescente, cioè

$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n$$

e tronchiamo la somma dopo r termini otteniamo una approssimazione di rango r della matrice A. Se in particolare la matrice A è una matrice ottenuta codificando i bit di un'immagine, si capisce che si può comprimere la foto utilizzando per $r < n$ invece di A la matrice

$$A_r = \sum_{k=1}^{r} \sigma_k u_k v_k^T.$$

Vediamone un implementazione in Matlab. Scarichiamo dal sito

```
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?
       objectId=4772&objectType=File
```

il file `compression.zip` e una volta decompresso, utilizziamo la routine `imcompr.m` per comprimere un'immagine di tipo bitmap, come ad esempio il gatto salvato in `gatto.bmp`. Lanciando dalla shell di Matlab (non funziona in Octave!) il comando

```
>> [im] = imcompr ('gatto.bmp',20,'gatto20.bmp');
>> [im] = imcompr ('gatto.bmp',50,'gatto50.bmp');
>> [im] = imcompr ('gatto.bmp',100,'gatto100.bmp');
```

Il risultato sono 3 foto dello stesso gatto, con diversi fattori di compressione, in quanto abbiamo usato valori di r diversi (r=20, 50, 100).

4 Facoltativo: Un esempio: approssimazione polinomiale.

Sia f una funzione reale e continua e siano x_i punti a due a due distinti appartenenti al dominio di f. Si cerca il polinomio

$$p_{n-1}(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

per cui lo scarto quadratico

$$\sum_{i=1}^{m} [p_{n-1}(x_i) - f(x_i)]^2.$$

Tale polinomio p_{n-1} è noto come *polinomio di miglior approssimazione* . Si prova che ció corrisponde a risolvere il problema sovradeterminato $Ax = b$ dove

$$A = \begin{pmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_m & \ldots & x_m^{n-1} \end{pmatrix}$$

Figure 1: Compressione di una foto via SVD.

e $b = (f(x_1), \dots, f(x_m))$, $x \in \mathscr{R}^m$. Tale idea è alla base dell'approssimazione fornita dalla toolbox `polyfit`.

References

[1] D. Bini, M. Capovani, O. Menchi, *Metodi numerici per l'algebra lineare*, Zanichelli, 1988.