1 Lezione del 04-04-25

Riprendiamo il discorso dei problemi ai minimi quadrati.

1.1 Esistenza delle soluzioni al sistema delle equazioni normali

Avevamo detto che di sistemi $Ax = b$ con $A \in \mathbb{C}^{m\times n}$, $m > n$, cioè *sovradeterminati*, che se la soluzione di $Ax = b$ non esiste, ergo $\text{rank}(A|b) > \text{rank}(A)$, potrebbe essere interessante cercare di risolvere il problema di ottimizzazione:

$$\min_{x\in\mathbb{C}^n} |Ax - b|_2 = \min_{x\in\mathbb{C}^n} |b - Ax|_2$$

dove il punto p_1 di minimo è lo stesso di quello del problema:

$$\min_{x\in\mathbb{C}^n} |Ax - b|_2^2$$

Avevamo quindi definito la funzione ϕ, assunta per semplicità $\psi : \mathbb{R}^n \to [0, +\infty)$:

$$\psi(x) = |b - Ax|_2^2 = |r(x)|_2^2 = \sum_{i=1}^{n} r_i(x)^2$$

A questo punto il minimo sarà semplicemente nel punto:

$$\begin{pmatrix} \frac{\partial\psi}{\partial x_1}(x) \\ \vdots \\ \frac{\partial\psi}{\partial x_n}(x) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

da cui avevamo visto ricavamo il sistema **lineare** (in quanto ψ è quadratica e quindi deriva ad una lineare):

$$A^T Ax = A^T b$$

Non lo abbiamo dimostrato formalmente, ma possiamo dire che lo stesso vale nel campo complesso, cioè con $x \in \mathbb{C}^n$:

$$A^H Ax = A^H b$$

prese le hermitiane invece delle trasposte.

Vediamo quindi che vale il seguente teorema:

Teorema 1.1: Esistenza delle soluzioni dei problemi ai minimi quadrati

Il sistema $A^T Ax = A^T b$ ammette soluzione $\forall A \in \mathbb{R}^{m\times n}$ e $\forall b \in \mathbb{C}^m$ (1). Inoltre, la soluzione è unica se e solo se $\text{rank}(A) = n$, $(m > n)$ (2).

Dimostriamo le due tesi in ordine.

1. La prima tesi si dimostra verificando che $A^T b \in I_m(A^T A)$ con I_m *immagine* o **spazio delle colonne** di $A^T A$, cioè verificando che è rispettata la condizione:

   $$\text{rank}(A^T A|A^T b) = \text{rank}(A^T A)$$

   di Rouché-Capelli, per cui esiste una soluzione. Abbiamo quindi:

   $$I_m(A^T A) = \{y \in \mathbb{C}^n \ : \ y = A^T Az, \quad z \in \mathbb{C}^n\}$$

Il caso buono è $A^T b \in I_m(A)$, cioè:

$$I_m(A) = \{y \in \mathbb{C}^n \ : \ y = Az, \quad z \in \mathbb{C}^n\}$$

così che:

$$A^T b = A^T A z \in I_m(A)$$

Questo però non è sempre vero. Possiamo allora dire che uno spazio vettoriale è sempre dato dalla somma dello spazio delle colonne di una trasformazione A e il corrispettivo spazio perpendicolare:

$$\mathbb{C}^m = I_m(A) \oplus I_m(A)^\perp$$

con lo spazio perpendicolare allo spazio delle colonne definito come:

$$I_m(A)^\perp = \{y : y^H z = 0, \quad \forall z \in I_m(A)\}$$

Potremo allora riscrivere b come la somma delle componenti *parallela* e perpendicolare:

$$b \in \mathbb{C}^m \implies b = b_1 + b_2, \quad \begin{cases} b_1 \in I_m(A) \\ b_2 \in I_m(A)^\perp \end{cases}$$

Prendiamo allora $A^T b$ come:

$$A^T b = A^T b_1 + A^T b_2$$

Si ha che $A^T b_1 \in I_m(A^T A)$ da quanto avevamo detto prima, mentre riguardo ad $A^T b_2$ si può dire:

$$A^T b_2 = \begin{pmatrix} a_1^T b_2 \\ \vdots \\ a_m^T b_2 \end{pmatrix} = 0$$

perchè $b_2 \in I_m(A)^\perp$ e le colonne di A stanno in $I_m(A)$, quindi la condizione di Rouché-Capelli è soddisfatta e la soluzione esiste. □

2. Per l'unicità, vogliamo verificare che $\det(A^T A) \neq 0$ soddisfatta la proprietà $\text{rank}(A) = n$. Abbiamo allora che dal teorema di Binet-Cauchy *generalizzato*:

$$\det(A^T A) = \sum_{[j]} \det(A^T_{[j]}) \cdot \det(A_{[j]}) = \sum_{[j]} |\det(A_{[j]})|^2$$

dove la sommatoria varia su tutte le possibili scelte di n indici in $\{1, ..., m\}$. Vogliamo allora mostrare che $\det(A^T A) \geq 0$, cioè che $\exists$ almeno una sottomatrice $A_{[j]}$ tale che $\det(A_{[j]}) \neq 0$. Questa esisterà sempre se $\text{rank}(A) = n$, e quindi anche l'ultima tesi del teorema è soddisfatta. □

Vediamo due esempi pratici per osservare il teorema in azione.

1. Prendiamo il sistema:

$$\begin{cases} 2x_1 - x_2 = 1 \\ -x_1 + x_2 = 0 \\ x_1 + 2x_2 = 1 \end{cases}$$

con $m = 3$ equazioni in $n = 2$ incognite. Avremo che le matrici A e b, e l'incognita x, saranno:

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 1 \\ 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

per il sistema $Ax = b$. Vediamo che il rango di A è 2, quindi dal teorema appena dimostrato la soluzione sarà unica. Calcoliamo allora le matrici A^TA e A^tb per il sistema $A^TAx = A^Tb$:

$$A^TA = \begin{pmatrix} 6 & -1 \\ -1 & 6 \end{pmatrix}, \quad A^Tb = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Avremo quindi il sistema:

$$\begin{cases} 6x_1 - x_2 = 3 \\ -x_1 + 6x_2 = 1 \end{cases}$$

da cui:

$$x^* = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{19}{35} \\ \frac{9}{35} \end{pmatrix}$$

Che è la soluzione ottima. Possiamo quindi calcolare la norma di residuo:

$$r = b - Ax^* = b - A \cdot \begin{pmatrix} \frac{19}{35} \\ \frac{9}{35} \end{pmatrix} = \begin{pmatrix} \frac{6}{35} \\ \frac{10}{35} \\ -\frac{2}{35} \end{pmatrix}$$

da cui ottieniamo uno scarto di ≈ 0.3381 dal valore ideale.

2. Un problema simmetrico al precedente potrebbe essere quello di trovare per quali parametri una matrice parametrizzata rispetta il teorema, cioè ammette una soluzione unica. Prendiamo ad esempio il sistema:

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & \alpha & 1 \\ 0 & 0 & \alpha \\ \beta & 0 & 0 \end{pmatrix}}_{A} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}}_{b}$$

con $\alpha, \beta \in \mathbb{R}$. Iniziamo imponendo l'antitesi del teorema cioè:

$$\text{rank}(A) < n \;\Leftrightarrow\; \exists A_{[j]} \in \mathbb{R}^{3\times 3} \text{ non invertibile}$$

con $A_{[j]}$ le sottomatrici quadrate 3×3 prese dal teorema di Binet-Cauchy. Consideriamo allora tutte le possibili $A_{[j]}$ coi loro determinanti:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & \alpha & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & \alpha & 1 \\ \beta & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & \alpha \\ \beta & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & \alpha & 1 \\ 0 & 0 & \alpha \\ \beta & 0 & 0 \end{pmatrix}$$
$$\downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow$$
$$\alpha(\alpha - 1) \qquad \beta(1 - \alpha) \qquad \alpha\beta \qquad \alpha^2\beta$$

da cui avere tutti determinanti nulli singifica soddisfare il sistema (non lineare):

$$\begin{cases} \alpha(\alpha - 1) = 0 \\ \beta(1 - \alpha) = 0 \\ \alpha\beta = 0 \\ \alpha^2\beta = 0 \end{cases}$$

con soluzioni $(\alpha, \beta) = (0, 0)$ o $(\alpha, \beta) = (1, 0)$. Si ha quindi che per queste due combinazioni di valori di α e β il sistema di partenza non ammette soluzione ottima unica.

1.1.1 Considerazioni di complessità del sistema delle equazioni normali

Valutiamo la complessità del sistema visto finora. Dobbiamo:

1. Calcolare $A^H A$, che ha complessità $O(mn^2)$ (da una $(n \times m)(m \times n)$);
2. Calcolare $A^H b$, che ha complessità $O(mn)$ (da una $(n \times m)(m \times 1)$);
3. Risolvere il sistema $A^H Ax = A^H b$, che ha complessità $O(mn^2 + n^3) \sim O(n^3)$.

Vediamo che il problema non è tanto la complessità, ma il fatto che $A^T A$ ha spesso un numero di condizionamento piuttosto alto, ad esempio nel caso $m = n$ si ha:

$$\mu(A^T A) = \mu(A)^2$$

Se $\mu(A^T A)$ ($< 10^k$ con k moderato), questo è fastidioso in quanto abbatte sostanzialmente la precisione. Conviene quindi applicare un metodo alternativo.

1.2 Metodo QR per problemi ai minimi quadrati

Vediamo quindi un metodo basato sulla fattorizzazione QR.

1.2.1 Fattorizzazione QR

La **fattorizzazione QR** è un tipo di fattorizzazione matriciale, come lo era la *fattorizzazione LU* che abbiamo già visto (8.3.4).

Diamone quindi la definizione:

Definizione 1.1: Fattorizzazione LU

Data $A \in \mathbb{C}^{m \times n}$, con $m \geq n$, una fattorizzazione QR di A è una coppia di matrici (Q, R) tali che: $A = QR$, con $Q \in \mathbb{C}^{m \times m}$ unitaria ($Q^H Q = QQ^H = I$), e $R \in \mathbb{C}^{m \times m}$ della forma triangolare superiore rettangolare (con un certo numero di righe a zero sotto il triangolo), cioè esprimibile come:

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

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

Varrà allora il teorema:

Teorema 1.2: Esistenza della fattorizzazione QR

Per ogni matrice $\exists$ sempre una fattorizzazione QR, ma non è mai unica.

Questo viene dal fatto che data $A = QR$, si potrebbe prendere $D \in \mathbb{R}^{m \times m}$ diagonale con $|d_j| = 1$ per $j = 1, ..., m$, e dire:

$$A = Q_2 R_2, \quad \begin{cases} Q_2 = QD \\ R_2 = D^{-1} R \end{cases}$$

in quanto:

$$A = QR = \underbrace{QD}_{Q_2} \cdot \underbrace{D^{-1} R}_{R_2}$$

Una proprietà di rilievo è che la norma 2 e la norma di Frobenius sono invarianti per moltiplicazioni con matrici unitarie, ovvero:

$$|A| = |QA| = |AQ| = |Q_1 A Q_2|$$

con norma $|\cdot|$ 2 o di Frobenius.

Dimostriamo per le due norme:

- **Norma 2:** questo si ha prendendo la definizione di norma 2:

  $$|A|_2 = \sqrt{\rho(A^H A)}$$

  Allora la norma di AQ, ad esempio, sarà:

  $$|AQ|_2 = \sqrt{\rho((AQ)^H AQ)} = \sqrt{\rho(Q^H A^H AQ)}$$

  $Q^H A^H AQ$ è simile a $A^H A$ in quanto dalla definizione di unarietà di Q, si può intendere $Q^H = Q^{-1}$ e quindi prendere la forma come un applicazione di matrice di trasformazione, da cui la tesi per la norma 2. □

- **Norma di Frobenius:** questo si ha prendendo la definizione di norma di Frobenius:

  $$|A|_F = \sqrt{\text{trace}(A^H A)}$$

  Allora la norma di AQ, ad esempio, sarà:

  $$|AQ|_F = \sqrt{\text{trace}(Q^H A^H AQ)}$$

  da cui come sopra. □

Nei problemi ai minimi quadrati, la fattorizzazione QR è utile in quanto se $A = QR$:

$$\min_{x \in \mathbb{C}^n} |b - Ax|_2 = \min_{x \in \mathbb{C}^n} |b - QRx|_2 = \min_{x \in \mathbb{C}} |Q^H b - Rx|_2$$

moltiplicando per Q^H all'ultimo passaggio.

Chiamato $Q^H b = c$, osserviamo quindi poter prendere:

$$\psi(x) = |cx - Rx|_2^2$$

che preso:

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

diventa:

$$\psi(x) = \left| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} R_1 \\ 0 \end{pmatrix} x \right|_2^2 = \left| \begin{pmatrix} c_1 - R_1 x \\ c_2 \end{pmatrix} \right|_2^2 = |c_1 - R_1 x|_2^2 + |c_2|_2^2$$

l'unico termine su cui abbiamo controllo è il primo, e possiamo quindi trascurare $|c_2|_2^2$. Prendiamo allora:

$$\psi'(x) = |c_1 - R_1 x|_2^2$$

che rappresenta un sistema lineare $n \times n$, dove bastera scegliere x tale che:

$$R_1 x = c$$

così da trovare 0 e ottenere il valore di ψ' (e quindi di ψ) minore possibile. Il vantaggio aggiunto è che R_1 è triangolare superiore, e quindi si può calcolare:

$$x = R_1^{-1} c$$

per sostituzione all'indietro.

Riassumendo, quindi, abbiamo ottenuto il metodo:

1. Si calcola la fattorizzazione $A = QR$;
2. Si calcola $Q^H b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$, che ha costo $O(nm)$ (non è $O(m^2 n)$, per via della struttura di Q);
3. Si risolve $R_1 x = c_1$, che è un sistema lineare triangolare superiore $n \times n$, quindi ha costo $O(n^2)$ per sostituzione all'indietro.

L'unica incognita è allora il costo della fattorizzazione, che anticipiamo è $O(mn^2)$.

1.2.2 Matrici di Householder

Una proprietà rilevante è che dato un vettore $v \in \mathbb{C}^m$ esiste sempre una matrice unitaria H_v tale che:

$$H_v \cdot v = c \cdot e_1, \quad c \in \mathbb{C}$$

per e_1 primo elemento della base canonica. Dato che la moltiplicazione per matrici unitarie non cambia la norma si ha:

$$|c| = |v|_2$$

Per creare tale matrice, scegliamo $\tilde{v}$:

$$\tilde{v} = v \pm |v|_2 \cdot e_1 = \begin{pmatrix} v_1 \pm |v|_2 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}$$

H_v dovrà quindi essere scelta come:

$$H_v = I - 2\frac{\tilde{v}\tilde{v}^H}{|\tilde{v}|_2^2}$$

dove il termine $\frac{\tilde{v}\tilde{v}^H}{|\tilde{v}|_2^2}$ rappresenta una proiezione sulla retta tracciata dal vettore $\tilde{v}$, e quindi H_v uguale alla *matrice di riflessione* sull'iperpiano ortogonale a $\tilde{v}$. Visto che abbiamo scelto $\tilde{v}$ perchè fosse effettivamente *"equidistante"* sia da v che da e_1, otteniamo una matrice che porta v esattamente sulla base e_1.

Le matrici costruite in questo modo vengono anche dette **Matrici di Householder** associate al vettore v.

Possiamo scrivere una funzione MATLAB, che ci torneà utile a breve, per il calcolo della matrice di Householder relativa a un certo vettore:

```
function H = householder(v)
    H = eye(length(v));
    vt = v;

    if vt(1) < 0
        vt(1) = vt(1) - norm(v); % si sottrae
    else
        vt(1) = vt(1) + norm(v);
    end

    H = H - 2 * vt * vt' / norm(vt)^2;
end
```

Vediamo inoltre come si è scelto il segno della norma aggiunta a `vt` ($\tilde{v}$) in modo da ridurre gli errori di cancellazione.

1.2.3 Calcolo della fattorizzazione QR

Analogamente al metodo di Gauss, per la fattorizzazione QR si moltiplica a sinistra per matrici unitarie che *"sistemano"* gli elementi sotto la diagonale.

Sfruttando la proprietà delle matrici di Householder, agiamo come segue:

$$H_1 = H_{a_1} = \text{matrice di Householder associata ad } a_1\text{, prima colonna di } A$$

diciamo:

$$H_1 \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} = \begin{pmatrix} ? & \dots \\ 0 & \dots \\ \vdots & \dots \\ 0 & \dots \end{pmatrix}$$

a questo punto basterà prendere H_2 per *"sistemare"* gli elementi della sottomatrice di coda successiva:

$$H_2 = \left(\begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & \tilde{H}_2 & \\ 0 & & & \end{array}\right)$$

e così via, introducendo di volta in volta più elementi della diagonale in testa. Si potrà quindi definire la matrice R come:

$$R = \underbrace{H_1^H H_2^H \dots H_n^H}_{Q} A$$

dove il prodotto delle unitarie rappresenterà la matrice Q.

1.2.4 Ottimizzazioni della fattorizzazione QR

Osserviamo che per avere costo $O(mn^2)$ (ed anche costo $O(mn)$ per $Q^H b$) si sfrutta il fatto che H_j è del tipo *identità + rango 1*, cioè:

$$H_j = I - bb^T$$

dove gli a e b vengono presi qui a scopo di esempio, risulterebbero dalla forma della matrice di Householder cha abbiamo visto alla 12.2.2. La parte importante è che il prodotto per una matrice generica B assume la forma:

$$H_j \cdot B = B - b \cdot b^T \cdot B$$

dove l'ultimo termine è il prodotto di una colonna per il prodotto di una riga per B. Quindi, nel caso B sia un vettore b, si hanno solo prodotti scalari di costo m, ergo:

$$H_j b \to O(m)$$

Se prendiamo il prodotto per tutte le matrici H_j, cioè per Q, abbiamo che questa cosa si ripete n volte e quindi il costo è $O(mn)$.

$$\implies H_n \cdot H_{n-1} \cdot ... \cdot H_1 b \to O(mn)$$

il costo $O(mn^2)$ a questo punto è immediato dal fatto che abbiamo semplicemente una dimensione n in più dalla matrice A.

Osserviamo quindi la versione della fattorizzazione QR, implementata in MATLAB, che effettua queste ottimizzazioni:

```
function [Q, A] = opt_qr_decomp(A)
    function [a, b] = householder_vec(v)
        vt = v;

        if vt(1) < 0
            vt(1) = vt(1) - norm(v); % si sottrae
        else
            vt(1) = vt(1) + norm(v);
        end

        a = vt;
        b = vt / norm(vt)^2 * 2;
    end

    function A = rank_one_l(A, a, b, i)
        A(i:end, :) = A(i:end, :) ...
                            - a * b' * A(i:end, :);
    end

    function A = rank_one_r(A, a, b, i)
        A(:, i:end) = A(:, i:end) ...
                            - A(:, i:end) * a * b';
    end

    n = height(A);

    Q = eye(n);
```

```
28
29      for i = 1:n
30          [a, b] = householder_vec(A(i:end, i));
31          A = rank_one_l(A, a, b, i);
32          Q = rank_one_r(Q, a, b, i);
33      end
34  end
```

Osserviamo poi che nel caso $m = n$ si può sfruttare la fattorizzazione QR per risolvere sistemi lineari, con costo $O(\frac{4}{3}n^3)$ per la fattorizzazione, il doppio rispetto a Gauss, ma solitamente più stabile.

Osserviamo infine che se $A \in \mathbb{R}^{m \times n}$ si può restare nei reali ed avere $Q \in \mathbb{R}^{m \times m}$, $Q^T Q = I$, ecc...

1.2.5 Implementazione MATLAB della decomposizione QR

Concludiamo quindi con l'implementazione MATLAB di questo algoritmo:

```
function [Q, A] = qr_decomp(A)
    n = height(A);
    Q = eye(n); % accumulatore

    for i = 1:n
        Hi = householder(A(i:end, i));
        Hi = blkdiag(eye(i - 1), Hi);
        A = Hi * A;
        Q = Q * Hi;
    end
end
```

Dove la funzione `blkdiag` ci permette di realizzare la forma a blocchi diagonale, con l'identità in testa, che la H_j assume ad ogni passaggio dell'algoritmo.