1 Lezione del 07-03-25

Proseguiamo i richiami di algebra lineare.

1.0.1 Approfondimento sulle prestazioni delle operazioni matriciali

Usiamo MATLAB per studiare il tempo di computazione richiesto per effettuare alcune delle operazioni che abbiamo visto.

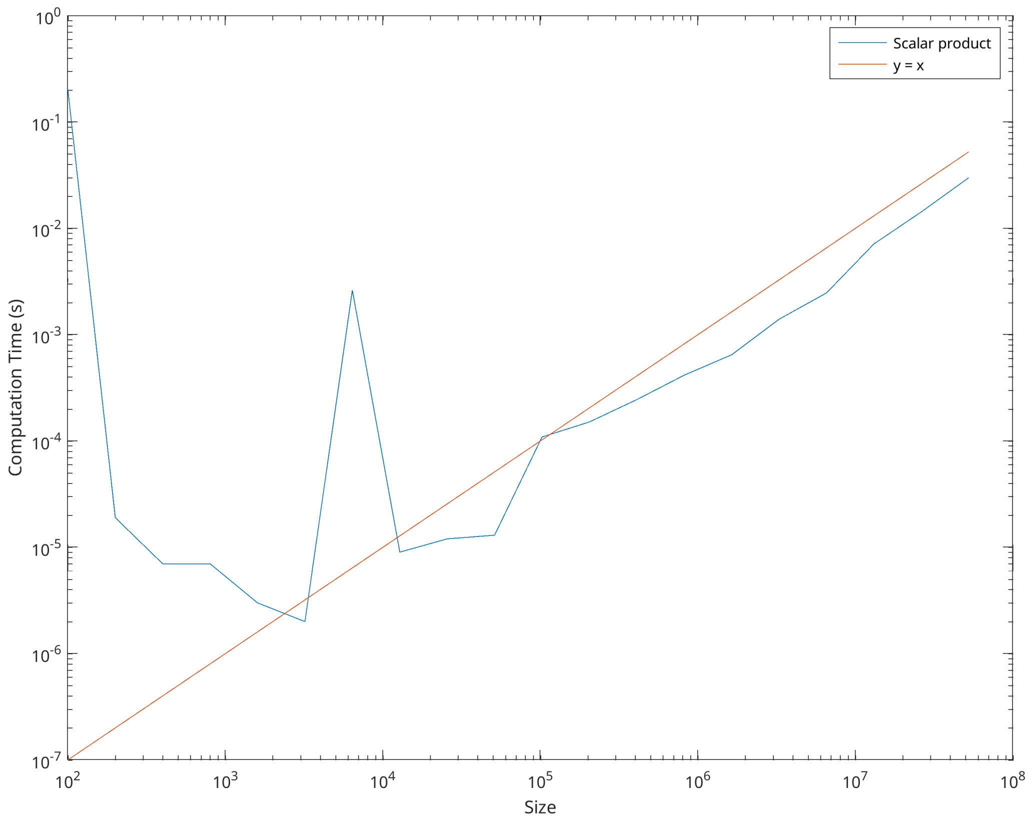
- Potremmo iniziare con il **prodotto scalare**. Avevamo detto che la complessità era $O(n)$. Scriviamo allora uno script per valutare, in scala logaritmica, il prodotto scalare fra vettori di dimensioni crescenti:

```
function n = time_scalar(num, base, mul)
    if nargin < 3
        mul = 100;
    end
    if nargin < 2
        base = 2;
    end

    n = zeros(num, 1);
    for i = 1:num
        n(i) = base^(i - 1) * mul;
    end

    times = zeros(num, 1);

    for i = 1:num
        A = randn(1, n(i));
        B = randn(n(i), 1);
        tic;
        C = A * B;
        times(i) = toc;
    end

    loglog(n, times);
    xlabel('Size');
    ylabel('Computation Time (s)');
end
```

  Un esempio di esecuzione potrebbe allora essere:

```
>> n = time_scalar(20);
>> hold on;
>> loglog(n, n * 10^-9);
>> hold off;
```

  dove si è cercato di fare il *fitting* della curva ottenuta con la funzione $y = x$ (dove la variabile è chiaramente n, e si è aggiunta una costante moltiplicativa k per avvicinarci di più al grafico originale). Il grafico ottenuto è quindi:



Vediamo dal grafico che le prestazioni sono effettivamente vicine a quelle previste dalla complessità computazionale, se non per errori dati all'overhead di inizializzazione del timer `tic` e `toc` di MATLAB (lato sinistro del grafico) e della velocità generale molto elevata delle operazioni, che rende più visibile il rumore dato dallo scheduling della CPU e altri fattori di basso livello.

- Vediamo allora l'andamento del **prodotto fra matrici**. Lo script MATLAB è praticamente identico al precedente:

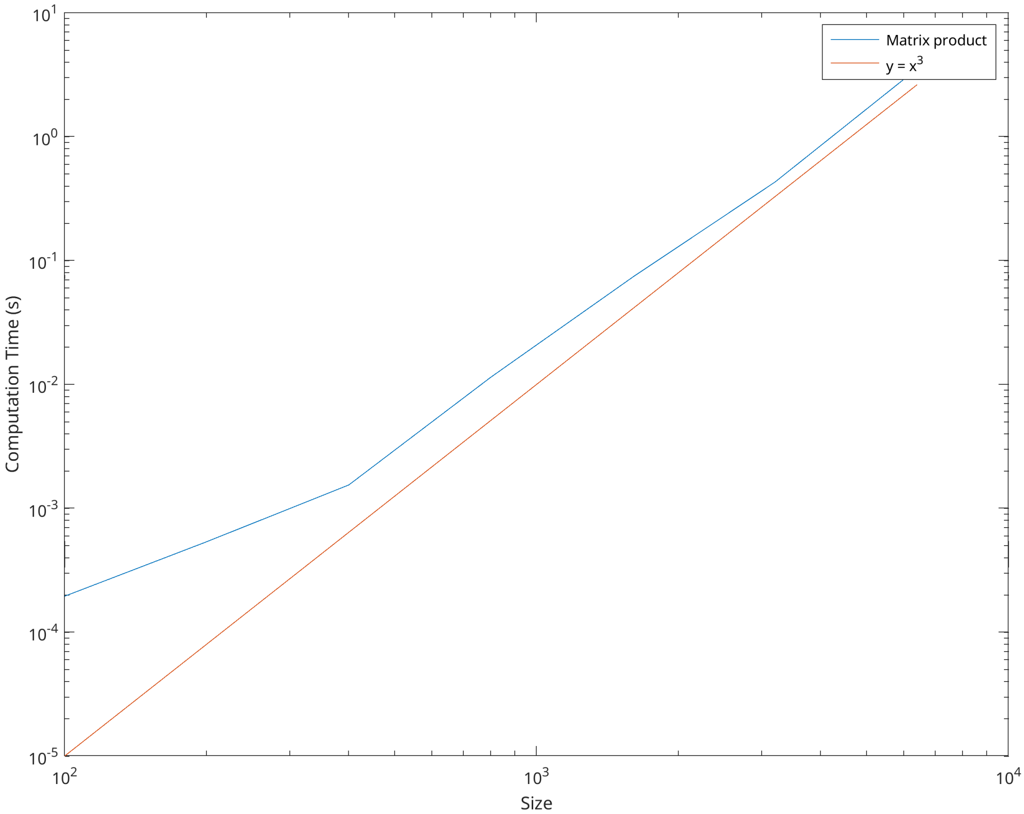
```
function n = time_matrix(num, base, mul)
    if nargin < 3
        mul = 100;
    end
    if nargin < 2
        base = 2;
    end

    n = zeros(num, 1);
    for i = 1:num
        n(i) = base^(i - 1) * mul;
    end

    times = zeros(num, 1);

    for i = 1:num
        A = randn(n(i));
        B = randn(n(i));
        tic;
        C = A * B;
        times(i) = toc;
    end
```

```

    loglog(n, times);
    xlabel('Size');
    ylabel('Computation Time (s)');
end
```

che eseguito come:

```
>> n = time_matrix(7);
>> hold on;
>> loglog(n, n.^3 * 10^-11);
>> hold off;
```

ci restituisce il grafico:



che notiamo è già più vicino del prodotto scalare (a causa del tempo di esecuzione maggiore richiesto, che "maschera" bene gli effetti a basso livello della CPU).

1.1 Proprietà del prodotto matriciale

Abbiamo che in genere il prodotto fra matrici non è **commutativo**, cioè:

$$AB \neq BA$$

di contro, vale l'**associativa**:

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

e la **distributiva**, separatamente ai due lati:

$$(A + B) \cdot C = A \cdot C + B \cdot C$$

$$C \cdot (A + B) = C \cdot A + C \cdot B$$

Inoltre, notiamo che quello delle matrici non è un *dominio integrale*:

$$A \cdot B = 0 \not\Longrightarrow A = 0 \lor B = 0$$

Vediamo ad esempio come sfruttare la proprietà associativa può permetterci di ottenere algoritmi più veloci. Supponiamo di voler calcolare:

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

con $A \in \mathbb{C}^{m \times n}$; $B \in \mathbb{C}^{n \times p}$, $C \in \mathbb{C}^{p \times q}$.
L'uguaglianza ci darà due metodi:

- $(A \cdot B) \cdot C$:
  - $A \cdot B = P \in \mathbb{C}^{m \times p}, \quad O(m \cdot n \cdot p)$
  - $P \cdot C = R \in \mathbb{C}^{m \times q}, \quad O(m \cdot p \cdot q)$

  $\implies O(mnp + mpq) = O\left(mp(n + q)\right)$
- $A \cdot (B \cdot C)$:
  - $B \cdot C = P \in \mathbb{C}^{n \times q}, \quad O(n \cdot p \cdot q)$
  - $A \cdot P = R \in \mathbb{C}^{m \times q}, \quad O(m \cdot n \cdot q)$

  $\implies O(npq + mnq) = O\left(nq(p + m)\right)$

che sono quindi uguali per $m = n = p$, ma (anche radicalmente diversi) diversi al variare di questi parametri.
In generale, quindi, se si hanno matrici con poche righe o poche colonne, è opportuno cercare di mantenere questa properietà più a lungo possibile nei risultati intermedi.

Possiamo fare considerazioni simili a quelle fatte sull'associativa con la distributiva. Ad esempio, posta l'uguaglianza:

$$(A + B) \cdot C = AC + BC$$

con $A, B \in \mathbb{C}^{m \times n}$ e $C \in \mathbb{C}^{n \times p}$, abbiamo due metodi:

- $(A + B) \cdot C$:
  - $A + B = P \in \mathbb{C}^{m \times n}, \quad O(m \cdot n)$
  - $P \cdot C = R \in \mathbb{C}^{m \times p}, \quad O(m \cdot n \cdot p)$

  $\implies O(mn + mnp) = O\left(mn(1 + p)\right)$
- $AC + BC$:
  - $A \cdot C = P_1 \in \mathbb{C}^{m \times p}, \quad O(m \cdot n \cdot p)$
  - $B \cdot C = P_2 \in \mathbb{C}^{m \times p}, \quad O(m \cdot n \cdot p)$

  $\implies O(mnp + mnp + mp) = O\left(mp(1 + 2n)\right)$

che sono sì asintoticamente uguali per $m = n = p$ ($O(n^3)$), ma non esattamente uguali in quanto da un lato si ha $O(n^2 + n^3)$ e dall'altro $O(n^2 + 2n^3)$. Questo risulta chiaramente dal fatto che col secondo metodo decidiamo di effettuare il prodotto matriciale (che è l'operazione più costosa) non una ma 2 volte.

Un altra proprietà del prodotto di matrici e che si può vedere il risultato riga per riga o colonna per colonna nei seguenti modi:

$$A = \begin{pmatrix} A_1 \\ \dots \\ A_m \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & \dots & B_p \end{pmatrix}$$

$$\implies A \cdot B = \begin{pmatrix} A \cdot B_1 & \dots & A \cdot B_p \end{pmatrix} = \begin{pmatrix} A_1 B \\ \dots \\ A_m B \end{pmatrix}$$

Questo ci dice che le colonne (le righe) di $C = A \times B$ sono combinazioni lineari delle colonne (delle righe) di A.

1.2 Determinante

Abbiamo visto la definizione di **determinante** per matrici quadrate:

Definizione 1.1: Determinante

Si definisce la funzione:

$$\det(A) : \mathbb{C}^{n \times n} \to \mathbb{C}$$

con:

$$\det(A) = \begin{cases} a_{11}, & n = 1 \\ a_{11}a_{22} - a_{12}a_{21}, & n = 2 \\ \sum_{j=1}^{n} (-1)^{i+j} \det(A_{ij}), & n > 2 \quad \text{(sviluppo di Laplace)} \end{cases}$$

determinante.

Osserviamo che il calcolo del determinante attraverso lo sviluppo di Laplace ha complessità algoritmica $O(n!)$.

1.2.1 Proprietà del determinante

Sappiamo che A invertibile $\Leftrightarrow$ $\det(A)$ (cioè A **singolare**). Inoltre valgono:

$$\det(A^T) = \det(A)$$

$$\det(A^H) = \overline{\det(A)}$$

1.3 Rango

Vediamo poi come ci può servire il determinante delle **sottomatrici**:

Definizione 1.2: Sottomatrice

Una sottomatrice di A è una matrice ottenuta prendendo la restrizione di A a un sottoinsieme di righe e di colonne, cioè data $A \in \mathbb{C}^{n \times n}$, $I, J \subseteq \{1, ..., n\}$ con $|I| = n_1$ e $|J| = n_2$, sarà allora:

$$A(I, J) \in \mathbb{C}^{n_1 \times n_2}$$

ottenuta incrociando le righe in I con le colonne in J.

Definiamo quindi i **minori**:

Definizione 1.3: Minore

Si dice minore di ordine k, con $k \in \{1, ..., n\}$ il determinante di una sottomatrice quadrata $k \times k$.

E le sottomatrici **principali** e **principali di testa**:

Definizione 1.4: Sottomatrice principale

Una sottomatrice si dice principale se gli insiemi I, J usati per estrarla sono $I = J$.

Definizione 1.5: Sottomatrice principale di testa

Una sottomatrice quadrata di ordine k si dice sottomatrice principale di testa se $I = J = \{1, ..., k\}$.

Allo stesso modo si possono definire **matrici di coda** (basterà prendere indici da k ad n).

Possiamo quindi definire il **rango** di matrice:

Definizione 1.6: Rango

Il rango $\mathrm{rank}(A)$ di una matrice A è definito come il massimo numero di colonne (o di righe) linearmente indipendenti, ed è uguale all'ordine massimo dei minori $\neq 0$ in A.

1.3.1 Proprietà del rango

Caso particolare del rango sarà chiaramente quello dove prendiamo come sottomatrice la matrice stessa: $\det(A) \neq 0 \Leftrightarrow \mathrm{rank}(A) = n$, e quindi gli insiemi delle colonne e delle righe di A sono tutte linearmente indipendenti.

Si ha poi che:

$$\mathrm{rank}(A) = \dim(\mathrm{Im}(A))$$

dove $\mathrm{Im}(A)$ è la dimensione dell'**immagine** di A:

$$\mathrm{Im}(A) = \{y \in \mathbb{C}^m : y = Ax, \ x \in \mathbb{C}^n\}$$

1.3.2 Teorema di Binet-Cauchy

Concludiamo enunciando il teorema di Binet-Cauchy sul determinante rispetto al prodotto matriciale.

Teorema 1.1: di Binet-Cauchy

Prese due matrici, $A \in \mathbb{C}^{n \times n}$ e $B \in \mathbb{C}^{n \times n}$, e $C = A \cdot B$ di dimensioni uguali, sarà:

$$\det(C) = \det(A) \cdot \det(B)$$

Nel caso generale avremo $A \in \mathbb{C}^{n \times p}$ e $B \in \mathbb{C}^{p \times n}$, da cui:

$$\det(C) = \begin{cases} 0, & p \leq n \\ \sum_j A_{[j]} \cdot B_{[j]} \end{cases}$$

dove $A_{[j]}$ e $B_{[j]}$ sono i minori di ordine n relativi ala stessa scelta di indici in A e in B.

La dimostrazione di questo teorema si basa sullo sviluppo dei cofattori della matrice $A \cdot B$, dove si possono riarrangiare i termini per ottenere gli sviluppi dei cofattori delle matrici A e B.

1.4 Matrice inversa

Diamo la definizione di **matrice inversa**:

Definizione 1.7: Matrice inversa

Data $A \in \mathbb{C}^{n \times n}$ tale che $\det(A) \neq 0$, si definisce $A^{-1} \in \mathbb{C}^{n \times n}$ tale che:

$$A \cdot A^{-1} = A^{-1} \cdot A = I$$

Se si guarda ad A come una funzione lineare $A : \mathbb{C}^n \to \mathbb{C}^n$, sarà che:

$$A : x \to Ax, \quad A^{-1} : Ax \to x$$

questo da:

$$\left(A^{-1} \circ A\right) = A^{-1}y = A^{-1}Ax = Ix = x$$

□

1.4.1 Proprietà della matrice inversa

Vediamo alcune proprietà di A^{-1}:

1. Ricordiamo di nuovo che A^{-1} esiste se e solo se $\det(A) \neq 0$, cioè equivalentemente se $\operatorname{rank}(A) = n$, o A ha spazi riga e colonna linearmente indipendenti;

2. Vediamo poi il calcolo di $\det(A^{-1})$: da $\det(I) = 1$, e:

$$\det(A \cdot A^{-1}) = \det(A) \cdot \det(A^{-1}), \quad \text{(Binet)}$$

$$\implies \det(A^{-1}) = \frac{1}{\det(A)}$$

3. Vediamo poi che:
$$(A \cdot B)^{-1} = B^{-1}A^{-1}$$
per matrici quadrate $A, B \in \mathbb{C}^{n \times n}$;

4. Riguardo alle trasposte e alle Hermitiane vale:
$$(A^T)^{-1} = (A^{-1})^T = A^{-T}$$
$$(A^H)^{-1} = (A^{-1})^H = A^{-H}$$
e:
$$(AB)^T = B^T A^T$$
$$(AB)^H = B^H A^H$$

1.5 Matrici particolari

Definiamo alcune matrici particolari:

Definizione 1.8: Matrici particolari

Data $A \in \mathbb{C}^{n \times n}$, si dice di A che è:

- **Hermitiana:** $A = A^H$;
- **Antiermitiana:** $A = -A^H$;
- **Unitaria** $A^H A = AA^H = I$, $A^{-1} = A^H$;
- **Normale** $A^H A = AA^H$;
- **Simmetrica** $A = A^T$;
- **Antisimmetrica** $A = -A^T$;
- **Ortogonale** $A^T A = AA^T = I$, $A^T = A^{-1}$

dove notiamo che **simmetrica** e **antisimmetrica** significano *hermitiana* e *antihermitiana* in $\mathbb{R}$, e **ortogonale** significa *unitaria* in $\mathbb{R}$.

Per di più vale:
$$\{\text{matrici simmetriche reali}\} \subseteq \{\text{matrici hermitiane}\} \subseteq \{\text{matrici normali}\}$$
e:
$$\{\text{matrici ortogonali}\} \subseteq \{\text{matrici unitarie}\}$$
dove notiamo che unitarie ed ortogonali hanno l'inversa "facile", nel senso che basta trasporre o trovare l'hermitiana.

1.6 Matrici di permutazione

Definiamo le **matrici di permutazione**:

Definizione 1.9: Matrice di permutazione

$A \in \mathbb{R}^{n \times n}$ si dice matrice di permutazione se A si ottiene dalla matrice di identità permutandone righe e colonne.

1.6.1 Proprietà delle matrici di permutazione

Tutte le matrici di permutazione sono matrici ortogonali (questo si vede dalla loro unimodularità, o osservando che P^T si ottiene dalla permutazione inversa).

Inoltre, vediamo che il prodotto di A con una matrice di permutazione si limita a scambiare righe e colonne in accordo con la permutazione della matrice. In particolare, $A \cdot P$ dà la permutazione delle colonne, mentre $P \cdot A$ dà la permutazione delle righe.

1.7 Sistemi lineari

Abbiamo un sistema di m equazioni lineari in n incognite:

$$\begin{cases} a_{11}x_1 + ...a_{1n}x_n = b_1 \\ ... \\ a_{m1}x_1 + ...a_{mn}x_n = b_m \end{cases}$$

Ci è spesso utile riscrivere sistemi di questo tipo in forma matriciale:

$$A = \begin{pmatrix} a_{11} & ... & a_{1n} \\ ... & ... & ... \\ a_{m1} & ... & a_{mn} \end{pmatrix} \in \mathbb{C}^{m \times n}, \quad x = \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix} \in \mathbb{C}^n, \quad b = \begin{pmatrix} b1 \\ ... \\ b_m \end{pmatrix} \in \mathbb{C}^m$$

Il problema principe sarà quello, date A e b, di trovare x.

1.7.1 Teorema di Rouché-Capelli

Richiamiamo il teorema:

Teorema 1.2: di Rouché-Capelli

$Ax = b$ ammette almeno una soluzione se $\operatorname{rank}(A) = \operatorname{rank}(A|b)$, con $(A|b) \in \mathbb{C}^{m \times (n+1)}$ la matrice ottenuta aumentando A con b.
Per quanto riguarda l'unicità, supposto $m \geq n$ (sistema *sovradeterminato*):

- Se $\operatorname{rank}(A) = n$ allora la soluzione è unica;
- Se $\operatorname{rank}(A) < n$ allora ci sono ∞ soluzioni, e l'insieme delle soluzioni forma uno spazio vettoriale affine di dimensione $n - \operatorname{rank}(A)$;

Nel caso il vettore $b = 0$, il sistema si dice **omogeneo**, e la soluzione nulla esiste sempre.

Abbiamo poi che:

Definizione 1.10:

L'insieme delle soluzioni di $Ax = 0$ si chiama Kernel (nucleo) della matrice:

$$\ker(A) = \{x \in \mathbb{R}^n : Ax = 0\}$$

notiamo che il kernel è un sottospazio vettoriale di dimensione $n - \operatorname{rank}(A)$.

Osserviamo infine che nel caso $m = n$, se $\det(A) \neq 0$, cioè $\operatorname{rank}(A) = n$, la soluzione di $Ax = b$ è unica per ogni $b \in \mathbb{R}^m$ e si scrive:

$$x = A^{-1}b$$

Vedremo che in generale calcolare l'inversa non è conveniente rispetto ad altri metodi di approssimazione.

1.7.2 Regola di Cramer

Il vettore soluzione si può scrivere ache componente per componente come:

$$x_j = \frac{\det(A_j)}{\det(A)}$$

dove A_j è la matrice ottenuta da A sotituendo la colonna j con b.

Questa via costa come calcolare $O(n)$ determinanti di matrici $n \times n$, ed è quindi poco conveniente ($O(n^3)$).