MATRICI e PRODOTTI

"speriamo che la troviate offensiva"

BLAS LEVEL 1 $x+y$, αx, --

Inner product $x, y \in \mathbb{R}^n$

$$\langle x, y \rangle = x^T y = \sum_{i=1}^{n} x_i y_i = [x_1 \cdots x_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Outer product $x \in \mathbb{R}^m$ $y \in \mathbb{R}^n$

$$x y^T = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} [y_1 \cdots y_n] = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_n \\ \vdots & & \vdots \\ x_m y_1 & \cdots & x_m y_n \end{bmatrix}$$

$$\text{rank}(x y^T) = 1$$

LEVEL 2

$A \in \mathbb{R}^{m \times n}$

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = [\underline{a}_1 \cdots \underline{a}_n] = \begin{bmatrix} \underline{a}^1 \\ \vdots \\ \underline{a}^m \end{bmatrix}$$

$\underline{a}_j \in \mathbb{R}^m$

$\underline{a}^i \in \mathbb{R}^n$

$y = Ax$ $x \in \mathbb{R}^n$ $y \in \mathbb{R}^m$

$$y_i = \sum_{j=1}^{n} a_{ij} x_j \qquad i = 1, \dots, m$$

Inner $y_i = \underline{a}^i \underline{x}$

$$y = \begin{bmatrix} \underline{a}^1 \underline{x} \\ \vdots \\ \underline{a}^m \underline{x} \end{bmatrix} = \begin{bmatrix} \underline{a}^1 \\ \vdots \\ \underline{a}^m \end{bmatrix} \underline{x}$$

Outer $y = [\underline{a}_1 \cdots \underline{a}_n] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \sum_{j=1}^{n} \underline{a}_j x_j$

Inner: il vettore y contiene "le proiezioni" di x lungo le righe di A

Outer: il vettore y è espresso come combinazione lineare delle colonne di A, usando come coefficienti le componenti di x.

In presenza di architetture parallele, tutte queste operazioni possono essere velocizzate, suddividendo il calcolo a blocchi.

Block-products

$$\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \underline{x} = \begin{bmatrix} A_1 \underline{x} \\ A_2 \underline{x} \end{bmatrix} \qquad \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = A_1 \underline{x}_1 + A_2 \underline{x}_2$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = \begin{bmatrix} A_{11}\underline{x}_1 + A_{12}\underline{x}_2 \\ A_{21}\underline{x}_1 + A_{22}\underline{x}_2 \end{bmatrix}$$

LEVEL 3

$$C = AB$$

$$(m \times q)\ (m \times n)(n \times q)$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1q} \\ \vdots & & \vdots \\ b_{n1} & \cdots & b_{nq} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1q} \\ \vdots & & \vdots \\ c_{m1} & \cdots & c_{mq} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$i = 1, \dots, m$

$j = 1, \dots, q$

Inner $\quad c_{ij} = \underline{a}^i \underline{b}_j = \langle \underline{a}^i, \underline{b}_j \rangle$

La matrice C contiene tutti i prodotti scalari tra le righe di A e le colonne di B.

$$C = \begin{bmatrix} \underline{a}^1 \\ \vdots \\ \underline{a}^m \end{bmatrix} \begin{bmatrix} \underline{b}_1 & \cdots & \underline{b}_q \end{bmatrix}$$

($C = A^T A$ contiene al suo interno la matrice di GRAM)

Outer

$$C = \begin{bmatrix} \underline{a}_1 & \cdots & \underline{a}_n \end{bmatrix} \begin{bmatrix} \underline{b}^1 \\ \vdots \\ \underline{b}^n \end{bmatrix} = \sum_{k=1}^{n} \underline{a}_k \underline{b}^k$$

La matrice C è decomposta come somma di n matrici di rango 1.

Per riga: $\underline{c}^i = \underline{a}^i B$ $\qquad$ **Per colonne:** $\underline{c}_j = A \underline{b}_j$

↳ Applicazione: calcolo dell'inversa A^{-1}.

Blocchi

$$\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \begin{bmatrix} B_1 & B_2 \end{bmatrix}, \quad \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Esempi $V = [\underline{v}_1 \; \cdots \; \underline{v}_n]$ matrice ortogonale

$V^T V = I \iff \underline{v}_i^T \underline{v}_j = \delta_{ij}$ base ortonormale

$c = V^T x$ proiezioni di x sui versori di base

$V c = V V^T x = x$ espressione di x nella base V

$W = [\underline{w}_1 \; \cdots \; \underline{w}_n]$ base non ortogonale

$W^{-1} = \begin{bmatrix} \underline{v}^1 \\ \vdots \\ \underline{v}^n \end{bmatrix}$ matrice inversa

$c = W^{-1} x$ coefficienti di x nella base W

$W c = W W^{-1} x = x$ rappresentazione di x nella base W

Altro modo per trovare i coefficienti:

risolvi $Wc = x$ $\left(x = \sum_{j=1}^{n} c_j \underline{w}_j \right)$

$c = W^{-1} x$ (stessa cosa!)

Altri prodotti

HADAMARD product (Schur product, dot product)

$$C = A \circ B \qquad c_{ij} = a_{ij} \cdot b_{ij} \qquad , A, B \in \mathbb{R}^{m \times n}$$

in Matlab $C = A .* B$, ma anche $A ./ B$, $A .\hat{} 2$, etc.

KRONECKER product (tensor product)

$$C = A \otimes B = \quad , \quad A \in \mathbb{R}^{m \times n} \quad B \in \mathbb{R}^{p \times q}$$

$$= \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp \times nq}$$

- tanti blocchi quanti sono gli elementi di A,
- ogni blocco grande quanto B.

in MATLAB: $C = kron(A, B)$

Generalizza l'outer product: $x \in \mathbb{R}^m \quad y \in \mathbb{R}^n$

$$x \otimes y^T = \begin{bmatrix} x_1 y \\ \vdots \\ x_m y \end{bmatrix} = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_n \\ \vdots & & \vdots \\ x_m y_1 & \cdots & x_m y_n \end{bmatrix} = x y^T$$

es.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 2 & 4 & 6 \\ 3 & 6 & 9 & 4 & 8 & 12 \end{bmatrix}$$

Es. autovalori/vettori di una matrice simmetrica (o normale)

$$A\underline{v}_i = \lambda_i \underline{v}_i \quad i=1,\dots,n \iff A[\underline{v}_1 \dots \underline{v}_n] = [\underline{v}_1 \dots \underline{v}_n]\begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_n\end{bmatrix}$$

da cui $\underline{AV = V\Lambda}$ vale per qualsiasi matrice quadrata.

Se $\exists V^{-1} \Rightarrow$ fattorizzazione spettrale $\Leftrightarrow$ matrice diagonalizzabile

Th spettrale A Hermitiana $\Rightarrow \lambda_i \in \mathbb{R}$, $\underline{v}_i \perp \underline{v}_j$ $i \neq j$

$\underline{v}_i \perp \underline{v}_j \iff V^T V = I$ (dimostra)

Si ottiene: $AV = V\Lambda \Rightarrow \underline{A = V\Lambda V^T}$ FATTORIZZAZIONE SPETTRALE (UNITARIA)

Rappresentazione outer product

$$V\Lambda V^T = [\underline{v}_1 \dots \underline{v}_n]\begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_n\end{bmatrix}\begin{bmatrix}\underline{v}_1^T \\ \vdots \\ \underline{v}_n^T\end{bmatrix} = [\lambda_1\underline{v}_1 \dots \lambda_n\underline{v}_n]\begin{bmatrix}\underline{v}_1^T \\ \vdots \\ \underline{v}_n^T\end{bmatrix} =$$

$$= \sum_{i=1}^{n} \lambda_i \underline{v}_i \underline{v}_i^T \quad \text{somma di matrici di rango 1.}$$

Applicazione: principal component analysis, dimensionality reduction, etc.

Supponiamo che A contenga dei dati e sia di grandi dimensioni.

Es. • matrice di incidenza di una rete (disegnata)

• qualsiasi tipo di dati memorizzati per colonne (in questo caso la simmetria può essere restrittiva).

Supponiamo anche che $\text{rank}_\epsilon(A) = k \ll n$, cioè

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_k| > \epsilon \geq |\lambda_{k+1}| \geq \dots \geq |\lambda_n|$$

Allora: $A = \sum_{i=1}^{n} \lambda_i \underline{v}_i \underline{v}_i^T \approx \sum_{i=1}^{k} \lambda_i \underline{v}_i \underline{v}_i^T = A_k$ **LOW RANK APPROXIMATION**

Compressione: $A \to \lambda_1 \dots \lambda_k, \underline{v}_1 \dots \underline{v}_k$

Calcolo: $f(A) \to f(\tilde{A})$ Es: A^n, e^A

Allora si può ottenere una LOW RANK APPROXIMATION.

$$A = \sum_{i=1}^{n} \lambda_i \underline{v}_i \underline{v}_i^T \approx \sum_{i=1}^{k} \lambda_i \underline{v}_i \underline{v}_i^T = A_k$$

$$A_k = [\underline{v}_1 \cdots \underline{v}_k] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} \begin{bmatrix} \underline{v}_1^T \\ \vdots \\ \underline{v}_k^T \end{bmatrix}$$

Compressione, Pattern recognition

Invece di memorizzare A (o processare A) si lavora su

$$\Lambda_k = \text{diag}(\lambda_1 \cdots \lambda_k), \quad V_k = [\underline{v}_1 \cdots \underline{v}_k]$$

Ricordiamo che $k \ll n$. Es: $n = 10000$, $k = 10$

A 10^8 numeri

A_k $\sim 10^5$ numeri

Calcolo

$f(A) \longrightarrow f(A_k)$ con A matrice simmetrica.

Es. $A^n = (V\Lambda V^T)^n = (V\Lambda V^T \cdot V\Lambda V^T \cdots V\Lambda V^T) = V\Lambda^n V^T$

questa formula è applicabile solo per dimensioni contenute. Se però $k \ll n$:

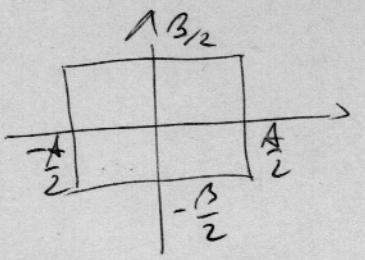
$$A^n \approx (A_k)^n = V_k \Lambda_k^n V_k^T$$

Per funzioni sviluppabili in serie, vale lo stesso. Es. $e^A = V e^{\Lambda} V^T$.

Questo approccio è utile perché esistono algoritmi in grado di calcolare alcuni autovalori/autovettori senza calcolarli tutti. (Arnoldi, Lanczos)

DATA SET SINTETICO PER PHOTOMETRIC STEREO

Sia $z = u(x,y)$, vogliamo costruire una "foto" della superficie.



$x \in \left[-\frac{A}{2}, \frac{A}{2}\right]$ $y \in \left[-\frac{B}{2}, \frac{B}{2}\right]$ $h = \frac{A}{r-1}$, $B = (s-1)h$

$x_i = -\frac{A}{2} + (i-1)h$, $i = 1, \dots, r$ Discretizzazione

$y_j = -\frac{B}{2} + (j-1)h$, $j = 1, \dots, s$

Legge di Lambert $I(x,y) = \rho(x,y) \cdot \langle \underline{n}(x,y), \underline{\ell} \rangle$

I intensità luminosa, ρ albedo, $\underline{n}$ vett normale, $\underline{\ell}$ vett. luce.

Sorgente all'infinito $\Rightarrow$ $\underline{\ell}$ uguale per tutti i punti

$$\underline{n}(x,y) = \frac{[-u_x, -u_y, 1]^T}{\sqrt{1+u_x^2+u_y^2}}$$ normalizzato.

$\underline{n}(x_i, y_j) = \underline{n}_{ij} = \underline{n}_k$, $k = j + (i-1)s$ ord. lessicografico

$k = 1, \dots, p = rs$

La lunghezza di $\underline{\ell}$ è proporzionale all'intensità della luce.

$N = [\underline{n}_1 \; \underline{n}_2 \; \dots \; \underline{n}_p]$ $3 \times p$ matrice delle normali

Supp. $\rho(x,y) = 1$

$\underline{m} = N^T \underline{\ell}$ è la foto coi pixel in ord. lessicografico → PRODOTTO MATRICIALE

$M = \text{reshape}(\underline{m}, r, s)$ è la foto.

La si può visualizzare o mappare sulle superficie (vedi `takephoto.m`)