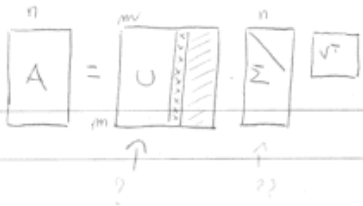
2 NOVEMBRE 2006

$A = U \Sigma V^T$



U e V sono ortogonali

$a_{ij} = \sum_{k=1}^{n} \sigma_k u_{ik} v_{jk}$

$a'_{ij} = \sum_{k=1}^{r} \sigma_k u_{ik} v_{jk}$

autovalore: peso su un asse ruotato (ellisse)

possiamo eliminare gli autovalori più piccoli

↓

le informazioni meno rilevanti

$A_r = U_r \Sigma_r V_r^T$

$(m,n) \quad (m,r) \quad (r,r) \quad (r,n)$

m parole
r concetti
? concetto viene espresso in parole

n documenti
r concetti
? documento

V ci dice il documento come sono espressi i concetti

$A = \underbrace{U^T U} \Sigma V^T$

I perché ortogonali

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

$\Sigma^{-1} U^T A = \underbrace{\Sigma^{-1} \Sigma} V^T$

I

$V^T = \Sigma^{-1} U^T A$

$V = (\Sigma^{-1} U^T A)^T = A^T U \Sigma^{-1}$

vale anche con $V_r = A_r^T U_r \Sigma_r^{-1}$

la prima riga di V è il prodotto tra la prima riga di A^T con il prodotto di $U_r \Sigma^{-1}$

V esprime i documenti come concetti

query

parole de ? a cercare

Le query le faccio in termini di parole .. ?

Le query al motore di ricerca le devo esprimere come concetti

moltp le query righe di A

$\hat{q} = q^T U \Sigma^{-1}$ · query espresse come concetti

$(1,1)\ (1,m)(m,m)\ (m,n)$

moltiplicando la query per $U_k \Sigma^{-1}$ → query come concetto

Bisogna adesso confrontarla con tutti i concetti.

esempio

$\underline{x} (x_1, \dots, x_p)$

$\underline{y} (y_1, \dots, y_p)$

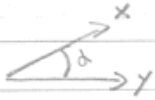
sono vicini? (I) un modo è fare la norma

?

$\sum (x_i - y_i)^2 = \|x-y\|$

(II)

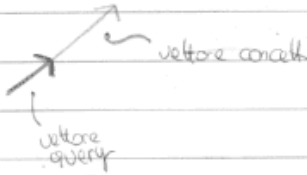
$\cos(\underline{x}, \underline{y})$



$\cos(x,y)$ $=1$ se coincidono ↑↑ $=-1$ ↓↑ $=0$ ↻

prodotto scalare

$(\underline{x} \cdot \underline{y}) = |x| \cdot |y| \cdot \cos(x,y)$



è importante l'angolo

query espresse nello spazio dei concetti

$\cos\Theta = \dfrac{\hat{q} \cdot \vec{v_j}}{\|\hat{q}\| \cdot \|v\|}$ cioè $\hat{q} = q^T \cdot U \cdot \Sigma^{-1}$

confrontare coseni

$\cos(\underline{q}, \underline{v})$ vicino a 1 è buono

coseno di v

QUADRATURA di GAUSS (stabile e convergente)

$$\int_a^b w(x) f(x)\,dx \cong \sum_{i=1}^{n} w_i f(x_i)$$

w_i pesi >0

x_i zeri del polinomio n-esimo polinomio ortogonale in $[a,b]$ rispetto a w

$[-1,1]$ $w(x)=1$ G-Legendre

$w_i(x) = \frac{1}{\sqrt{1-x^2}}$ Cherbicev 1° specie $w_i = \frac{\pi}{n}$

$w(x)=\sqrt{1-x^2}$ Cherbicev 2° specie

$w(x) = e^{-x^2}$ Hermite $(-\infty,\infty)$

$w(x)=e^{-x}$ Laguerre $(0,\infty)$

Formula di Gauss stabile perché $w_i>0$

convergenti: $\left(\sum_{i=1}^{n} w_i f(x_i) \to \int_a^b w(x) f(x)\phi(x)\right)$

Grado di precisione $2n-1$

$$f \in \mathbb{P}_{2n-1} \quad \int_a^b w(x) f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i)$$

(valore esatto)

Se f è smooth abbastanza (cioè simile ad un polinomio) allora Gauss approssima bene l'integrale

Su questa idea è basato QUADPACK

Per valutare in modo automatico $\int_a^b w(x) f(x)\,dx$ con f complessa (strategia consiste nell'addensare nodi dove f è meno regolare perché dove è più regolare può essere bene approssimata con pochi punti con Gauss)

es: $f(x) = x^5 \int_{-1}^{1} \frac{x^5}{\sqrt{1-x^2}}\,dx \cong \sum_{i=1}^{n} w_i f(x_i) = \sum \frac{\pi}{n} f\left(\cos\frac{\pi(2n-1)}{2n}\right)$

f inoltre è potenza di polinomi $2n-1$ . . . ?

f è x^5 cioè polinomi di grado $2n-1$ $(2\cdot 3-1)$ -> ho un valore esatto $\int_{-1}^{1} \frac{x^5}{\sqrt{1-x^2}} = \frac{\pi}{3}\sum_{i=1}^{3} \cos(\ldots)^5$

6 DICEMBRE 2006

$$f_n(x) = \sum_{k=1}^{n} c_k G_k(x)$$

G_k funzione continua in $[a,b]$

sistema ortonormale

$$\mu_n = \sqrt{\int_a^b \left\{ f(x) - \sum_{k=1}^{n} c_k G_k(x) \right\}^2 dx}$$

errore in norma quadratica. MINIMO

se $c_k = a_k = \int_a^b f(x) G_k(x)\, dx$ coeff Fourier

sappiamo l'esistenza della migliore approssimazione ma non sappiamo costruirla

"quasi" migliore approssimazione uniforme

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} \left\{ f(x) - f_n(x) \right\}^2 dx$$

$[-1,1]$ $\frac{1}{\sqrt{1-x^2}}$ Cebicev 1ª specie

f_n uso polinomi di Cebicev ortonormali

$$f_n(x) = a_0 \frac{T_0(x)}{\sqrt{\pi}} + \sum_{k=1}^{n} a_k \frac{T_k(x)}{\sqrt{\pi}} \frac{\sqrt{2}}{\sqrt{\pi}}$$

"quasi" approssimazione uniforme migliore

$$a_0 = \frac{1}{\sqrt{\pi}} \int_{-1}^{1} f(x) T_0(x)\, dx$$

$$a_k = \frac{\sqrt{2}}{\sqrt{\pi}} \int_{-1}^{1} f(x) T_k(x)\, dx$$

METODI PASSO-PASSO

EQUAZIONI DIFFERENZIALI

FORMULA $$y_{i+k} = \sum_{j=0}^{k-1} \left\{ a_j y_{i+j} + h b_j f(x_{i+j}, y_{i+j}) \right\} + h b_k f(x_{i+k}, y_{i+k})$$

autoportante — non autoportante

include eulero e eulero-cauchy

$b_k = 0$

metodo esplicito o predictor

$x_i \quad x_{i+1} \dots x_{i-k-1} \quad x_{i+k}$

$b_k \neq 0$ metodi impliciti (corrector)

es: MILNE

$$y_{i+1} = y_{i-3} + \frac{4}{3} h \left[2y'_{i-2} - y'_{i-1} + 2y'_i \right]$$

$$y_{i+1} = y_{i-1} + \frac{h}{3} \left(y'_{i-1} + h y'_i + y'_{i+1} \right)$$

$$y(x_{i+1}) = f\left(x_{i+1}, y(x_{i+1}) \right)$$

METODO RUNGE-KUTTA

$$y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2)$$

$$k_1 = h f(x_i, y_i)$$

$$k_2 = h f(x_i + h, y_i + k_1)$$

7 DICEMBRE 2006 www.na.iac.cnr.it/~bdv

esempio trasformata veloce di fourier

$$\begin{cases} c_j = \frac{1}{N}\sum_{k=0}^{n-1} f_k \exp\left(\frac{-i\,2\pi jk}{N}\right) \\ f_k = \sum_{j=0}^{n-1} c_j \exp\left(\underbrace{i\,\frac{2\pi jk}{N}}_{*}\right) \end{cases}$$

$$\begin{bmatrix} f_0^{(0)} \\ f_1^{(0)} \\ f_2^{(0)} \\ f_3^{(0)} \\ f_4^{(0)} \\ f_5^{(0)} \\ f_6^{(0)} \\ f_7^{(0)} \end{bmatrix} \rightarrow \begin{bmatrix} f_0^{(0)} + f_4^{(0)} \\ f_1^{(0)} + f_5^{(0)} \\ f_2^{(0)} + f_6^{(0)} \\ f_3^{(0)} + f_7^{(0)} \\ \hline f_0^{(0)} - f_4^{(0)} \\ (f_1^{(0)} - f_5^{(0)})\,\omega \\ (f_2^{(0)} - f_6^{(0)})\,\omega^2 \\ (f_3^{(0)} - f_7^{(0)})\,\omega^3 \end{bmatrix} =$$

$$= \begin{bmatrix} f_0^{(1)} \\ f_1^{(1)} \\ f_2^{(1)} \\ f_3^{(1)} \\ \hline f_4^{(1)} \\ f_5^{(1)} \\ f_6^{(1)} \\ f_7^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} f_0^{(1)} + f_2^{(1)} \\ f_1^{(1)} + f_3^{(1)} \\ f_0^{(1)} - f_2^{(1)} \\ (f_1^{(1)} - f_3^{(1)})\,\omega_1 \\ \hline f_4^{(1)} + f_6^{(1)} \\ f_5^{(1)} + f_7^{(1)} \\ f_4^{(1)} - f_6^{(1)} \\ (f_5^{(1)} - f_7^{(1)})\,\omega_1 \end{bmatrix} = \begin{bmatrix} f_0^{(2)} \\ f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \\ f_4^{(2)} \\ f_5^{(2)} \\ f_6^{(2)} \\ f_7^{(2)} \end{bmatrix} \rightarrow \begin{bmatrix} f_0^{(2)} + f_1^{(2)} \\ f_0^{(2)} - f_1^{(2)} \\ f_2^{(2)} + f_3^{(2)} \\ f_2^{(2)} - f_3^{(2)} \\ f_4^{(2)} + f_5^{(2)} \\ f_4^{(2)} - f_5^{(2)} \\ f_6^{(2)} + f_7^{(2)} \\ f_6^{(2)} - f_7^{(2)} \end{bmatrix} = \begin{bmatrix} f_0^{(3)} \\ f_1^{(3)} \\ f_2^{(3)} \\ f_3^{(3)} \\ f_4^{(3)} \\ f_5^{(3)} \\ f_6^{(3)} \\ f_7^{(3)} \end{bmatrix}$$

8 somme su 8 termini → 8 somme su 4 termini → 8 somme su 2 termini → 8 somme su un termine

$$\frac{1}{N} = \underbrace{\frac{1}{2}}_{\text{trascuriamo } \frac{1}{2}} \frac{1}{N/2}$$

$$C_{2m+1} = \frac{1}{2}\left[\frac{1}{N/2}\sum_{k=0}^{N/2-1} f_k^{(1)}\,\omega^{m_1 k}\right]$$

$$\frac{1}{2}\log_2 N \quad \frac{1}{N}$$

$$C_j = \frac{1}{N}\sum_{k=0}^{n-1} f_k\,\omega^{jk}$$

$$\left(\frac{1}{2}\right)^3 = \frac{1}{8}$$

ad ogni passo perdiamo $\frac{1}{2}$

$\frac{1}{2}$ × il numero di passi

inoltre si perde l'ordine per risalire all'ordine:

0	000	000	0	$f_0^{(3)} \rightarrow C_0$
1	001	100	4	$f_1^{(3)} \rightarrow C_4$
2	010	010	2	$f_2^{(3)} \rightarrow C_2$
3	011	110	6	$f_3^{(3)} \rightarrow C_6$
4	100	001	1	$f_4^{(3)} \rightarrow C_1$
5	101	101	5	$f_5^{(3)} \rightarrow C_5$
6	110	011	3	$f_6^{(3)} \rightarrow C_3$
7	111	111	7	$f_7^{(3)} \rightarrow C_7$

trasformata di Fourier bidimensionale

numero di pixel potenza di 2

N_1 →

N_2

ANALISI di FOURIER
(TRASFORMATA DIRETTA)

calcoliamo la trasf di Fourier x ogni riga e sostituiamo i coeff di ogni riga

calcoliamo la trasformata x ogni colonna, sostituendo i coeffic.

SINTESI stesso algoritmo (INVERSA) cambiando +i con −i

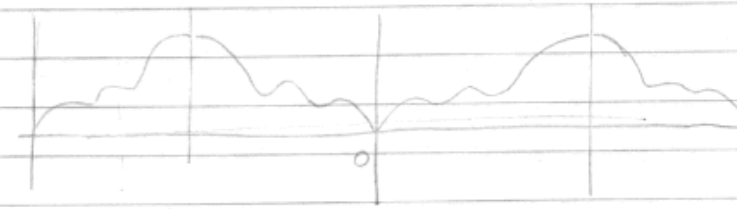
$IFT(x) = \overline{FT(\bar{x})}$ parte immaginaria cambiata di segno

In alcuni punti la trasformata di Fourier è reale

$$\exp(x) = \cos x + i \sin x$$

scompare la parte immaginaria

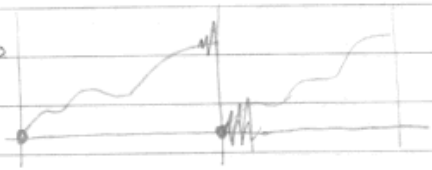
se raddoppiamo la funzione → funzione pari



Raddoppiare il dominio e riflettere la funzione data

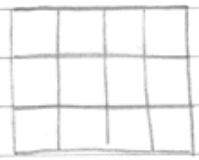
VANTAGGI:

- compressione efficace: valore all'estremo destra = valore estremo sinistra
- coeff. $\mathbb{R}$



C_0 C_4 C_2 C_6 C_1 C_5 C_3 C_7

x immagini con dimensioni non potenze di 2:



ogni blocco ha dimensioni potenze di 2 si applica la trasformata coseno ad ogni blocco

14 DICEMBRE 2006

$$P_N(x) = x^N + a_1 x^{N-1} + \dots + a_{N-1} x + a_N$$

$r = \max |a_i|$

radici di $P_N \in \mathbb{C} := \{ z \in \mathbb{C} / |z| \leq 1 + r \}$

calcolo di P_N in x N addizioni $2N-2$ moltiplicazioni

ALGORITMO di HORNER costo quasi dimezzato N addizioni $N-1$ moltiplicazioni

$$\begin{cases} P_0 = 1 \\ P_k = P_{k-1} x + a_k, \quad k = 1, \dots, N \\ P_N = P_N(x) \end{cases}$$

$$P_3(x) = x^3 + a_1 x^2 + a_2 x + a_3 = a_3 + x(a_2 + x(a_1 + x1))$$

$N = 3$

$P_0 = 1$
$P_1 = 1x + a_1$
$P_2 = (x + a_1)x + a_2$
$P_3 = P_2 x + a_3 = x((x + a_1)x + a_2) + a_3$

$$P_N(x_i) = \frac{P_N(x) - P_N(x_i)}{x - x_i}\Big|_{x = x_i} = q_{N-1}(x_i)$$

$$q_{N-1}(x) = \frac{P_N(x) - P_N(x_i)}{x - x_i} = x^{N-1} + b_1 x^{N-2} + \dots + b_{N-2} x + b_{N-1}$$

$b_0 = 1$

$b_k = b_{k-1} x_i + a_k, \quad k = 1, 2, \dots, N-1$

$x_1^{(0)}$ approssimaz. per x_1 radice P_N

valuto x_1 usando Newton

buona approssimazione

$P_N(x_1^{(n)})$ e $P_N(x_1^{(n)})$ x_1! $x_2^{(0)}$? per le altre radici?

$$q_{N-1}(x) = \frac{P_N(x) - P_N(x_1)}{x - x_1}$$

radici $x_2 \dots x_n$

calcolo x_2 con Newton

$x_2^{(0)}$ x_2?

radice q_{N-1}

Newton x_2! x_3! $x_3^{(0)}$

$$t_{N-2}(x) = \frac{q_{N-1}(x) - q_{N-1}(x_2)}{x - x_2}$$

$x_3 \dots x_n$

quindi

da approssimazioni grossolane trovo valori ben approssimati

poiché l'errore si accumula, lavorare con

$x_1^{(0)}$ $x_2^{(0)}$ ..... $x_N^{(0)}$ → max precisione possibile

x_1 x_2 ..... x_N → valutare radici in ordine crescente

$|x_1| \leq |x_2| \leq |x_3| \leq \dots$

→ i valori trovati vanno raffinati utilizzandoli come punto di input per una iterazione di Newton