Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

Decomposizioni tensoriali: High Order SVD e Canonical Decomposition

Tesi di Laurea in Calcolo Numerico

Relatore:
Chiar.ma Prof.ssa
VALERIA SIMONCINI

Presentata da:
FRANCESCO PALTRINIERI

Sessione II
Anno Accademico 2018/2019

Indice

Introduzione

Nel corso degli ultimi anni sempre più problemi di statistica, elaborazioni di segnali e modelli associati ad equazioni stocastiche hanno richiesto l'utilizzo di strutture matematiche di tre o più dimensioni per poter essere risolti. Queste strutture vengono chiamate tensori e possono essere visualizzati come array multidimensionali, una sorta di generalizzazione del concetto di vettori e matrici, che possono essere intese come tensori di ordine rispettivamente due e tre.

Questa tesi ha come obbiettivo quello di presentare questo strumento, sia dal punto di vista teorico, sia dal punto di vista applicativo. In particolare il primo capitolo sarà dedicato ad una introduzione formale dei tensori e delle loro operazioni fondamentali (quali ad esempio unfolding, prodotto mode-n e rango), successivamente daremo una panoramica di prodotti matriciali di Kronecker, Khatri-Rao e Hadamard. Nei successivi capitoli affronteremo due delle decomposizioni tensoriali più diffuse:

- High Order SVD
- Canonical decomposition

La prima è basata sulla Singular Value Decomposition delle matrici, e può essere considerata una sua estensione a più dimensioni. Mentre la seconda è strettamente associata alla definizione di rango tensoriale.

Per ogni parte, dopo una prima introduzione della decomposizione, viene trattata l'unicità ed un suo possibile utilizzo per approssimare il tensore iniziale. Successivamente viene presentato l'algoritmo e la sua dimostrazione di convergenza.

Ogni sezione viene conclusa con un'applicazione dell'algoritmo trattato. Nella prima, viene utilizzato il database *The Yale Face Database* presente alla pagina www.facerec.org/databases/. Viene analizzato cosa succede al tensore contenente tutte le immagini se sottoposto a una HOSVD troncata, scegliendo a ogni passo un numero diverso di autovettori. In particolare, viene studiato l'aumento del rumore che si viene a generare sulle immagini e successivamente si controlla la crescita degli errori di un algoritmo di riconoscimento. Per la seconda decomposizione viene invece studiato il grafico del residuo al variare del rango R considerato nell'approssimazione; in seguito viene discusso un esempio numerico di dimensioni ridotte.

Capitolo 1

I Tensori

Computazionalmente si può identificare un tensore con un array multidimensionale. Più formalmente un tensore di ordine N, $\mathcal{X}$ è un elemento del prodotto tensoriale di N copie di $\mathbb{R}$ ognuna col proprio sistema di coordinate: $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$.
I tensori estendono quindi il concetto di vettore e matrice, che possono essere interpretati come tensori di primo e secondo ordine, a più dimensioni.

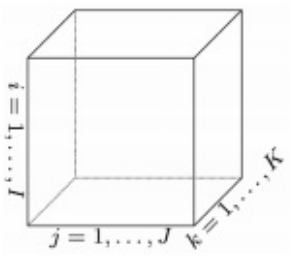

Figura 1.1: Una possibile rappresentazione di $\mathcal{X} \in \mathbb{R}^I \times \mathbb{R}^J \times \mathbb{R}^K$

È possibile definire una norma sullo spazio dei tensori:

Definizione 1.0.1. Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ allora la norma di $\mathcal{X}$ é

$$\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1,\cdots i_N}^2}.$$

Si osserva che questa norma è equivalente alla norma di Frobenius di $\mathbb{R}^{I_1 I_2 \cdots I_N}$.
Inoltre è ben definito il prodotto interno tra tensori:

Definizione 1.0.2. Sia $\mathcal{X}$ e $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ allora il prodotto scalare tra due tensori è:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1,\cdots i_N} y_{i_1,\cdots i_N}.$$

Di conseguenza si può definire l'ortogonalità tra due tensori, nel caso in cui il loro prodotto scalare sia uguale a 0.

Ovviamente $\langle \mathcal{X}, \mathcal{X} \rangle = \|\mathcal{X}\|^2$.

1.1 Unfolding

Unfolding o Matricization è un processo di riordino degli elementi di un tensore di N-esimo ordine in una matrice. Esistono diversi procedimenti per ottenerlo, di seguito viene trattato solo il mode-n unfolding, dato che è quello sul quale si basa l'HOSVD. Tutti i diversi unfolding possono essere ricavati l'uno dall'altro mediante la moltiplicazione per una matrice di permutazione. Quindi l'utilizzo di un metodo piuttosto che di un altro in un algoritmo, conduce alla stessa soluzione in entrambi i casi.

Definizione 1.1.1. Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ allora il suo mode-n è la matrice $X_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_N}$ costruita tale che l'elemento tensoriale $(i_1, i_2, \ldots, i_N)$ viene mappato nell'elemento matriciale (i_n, j) tale che

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k, \quad \text{con} \quad J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m.$$

Quindi la matrice $X_{(n)}$ è quella ottenuta considerando i vettori lungo il mode n (quindi fissando tutti gli indici tranne l'n-esimo) come colonne.

Esempio. Sia $\mathcal{X} \in \mathbb{R}^{3 \times 4 \times 2}$ tale che:

$$\mathcal{X}(:,:,1) = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \qquad \mathcal{X}(:,:,2) = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}.$$

Si hanno i seguenti tre diversi mode-n per il tensore $\mathcal{X}$:

$$X_{(1)} = \begin{bmatrix} 1 & 4 & \cdots & 19 & 22 \\ 2 & 5 & \cdots & 20 & 23 \\ 3 & 6 & \cdots & 21 & 24 \end{bmatrix},$$

$$X_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}, \qquad X_{(3)} = \begin{bmatrix} 1 & 2 & \cdots & 11 & 12 \\ 13 & 14 & \cdots & 23 & 24 \end{bmatrix}.$$

È inoltre possibile effettuare un vectorize del tensore ottenendo così un vettore, anzichè una matrice. Sia $\mathcal{X}$ come nell'esempio. Allora

$$\text{vec}(\mathcal{X}) = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 24 \end{bmatrix}.$$

1.2 Il prodotto Mode-n

A partire dalla definizione di mode-n di un tensore è possibile costruire il prodotto tra un tensore e una matrice.

Definizione 1.2.1. Sia $\mathcal{X}$ e $U \in \mathbb{R}^{J \times I_n}$ allora il prodotto mode-n tra $\mathcal{X}$ e U viene indicato con $(\mathcal{X} \times_n U)$ ed è il tensore di dimensione $I_1 \times \cdots I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ tale che elemento per elemento si ha:

$$(\mathcal{X} \times_n U)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j i_n}.$$

Risulta più immediato scritto in forma matriciale:

$$\mathcal{Y} = \mathcal{X} \times_n U \Leftrightarrow Y_{(n)} = U X_{(n)}.$$

Si osserva che per mode differenti l'ordine è irrilevante e vale quindi:

$$\mathcal{X} \times_m A \times_n B = \mathcal{X} \times_n B \times_m A \quad (m \neq n).$$

Mentre per uno stesso mode vale $\mathcal{X} \times_n A \times_n B = \mathcal{X} \times_n (AB)$.

1.3 Prodotti matriciali

Vi sono alcuni prodotti matriciali che risultano utili per lo studio degli algoritmi proposti, questi sono: il prodotto di Kronecker, il prodotto di Khatri-Rao e il prodotto di Hadamard.

Definizione 1.3.1. Siano $A \in \mathbb{R}^{I\times J}$ e $B \in \mathbb{R}^{K\times L}$. Allora il prodotto di Kronecker viene indicato con $A \otimes B$ ed è la matrice di dimensione $IK \times JL$ definita come segue:

$$\begin{aligned} A \otimes B &= \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix} \\ &= \begin{bmatrix} a_1 \otimes b_1 & a_1 \otimes b_2 & \cdots & a_J \otimes b_L \end{bmatrix}. \end{aligned}$$

Il prodotto di Khatri-Rao è definito a partire dal precedente, tuttavia necessita dell'ipotesi aggiuntiva che le due matrici abbiano lo stesso numero di colonne:

Definizione 1.3.2. Siano $A \in \mathbb{R}^{I\times K}$ e $B \in \mathbb{R}^{J\times K}$. Allora il prodotto di Khatri-Rao, denotato con $A \odot B$, è la matrice di dimensione $(IJ) \times K$ tale che ogni sua n-esima colonna sia il prodotto di Kronecker della n-esima colonna di A e B:

$$A \odot B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \cdots & a_K \otimes b_{k.} \end{bmatrix}$$

Definizione 1.3.3. Siano $A, B \in \mathbb{R}^{I\times J}$. Si definisce prodotto di Hadamard, $A * B$, la matrice di dimensione $I \times J$ costruita in questo modo:

$$A * B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}.$$

Esiste una relazione che collega il prodotto mode-n con il prodotto di Kronecker:

$$\begin{gathered} \mathcal{Y} = \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_N A^{(N)} \\ \Leftrightarrow \\ Y_{(n)} = A^{(n)} \mathcal{X}_{(n)} (A^{(N)} \otimes \cdots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)})^T. \end{gathered} \tag{1.1}$$

Un'importante proprietà, utilizzata più avanti, che combina gli altri prodotti è la seguente:

$$(A \odot B)^\dagger = ((A^T A) * (B^T B))^\dagger (A \odot B)^T, \tag{1.2}$$

dove $A^\dagger$ denota la matrice pseudoinversa di Moore-Penrose di A.

1.4 Rango tensoriale

Il concetto di rango matriciale viene esteso anche ai tensori, ma risulta essere leggermente diverso. Prima occorre definire un tensore di rango 1:

Definizione 1.4.1. Il tensore $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ si definisce di rango 1 se è ottenuto come prodotto esterno di N vettori, ovvero:

$$\mathcal{X} = a^{(1)} \circ a^{(2)} \cdots \circ a^{(N)},$$

cioè tale che $x_{i_1,i_2,\ldots,i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}$ per tutti $1 \leq i_n \leq I_N$.

Definizione 1.4.2. Il rango di un tensore $\mathcal{X}$ viene definito come il più piccolo numero di tensori di rango 1 che lo generano come loro somma.

Per esempio, sia $\mathcal{X}$ come in precedenza di rango R, allora si ha:

$$\mathcal{X} = \sum_{r=1}^{R} a^{(1)}_r \circ a^{(2)}_r \circ \cdots \circ a^{(N)}_r \quad \text{con } a^{(n)}_r \in \mathbb{R}^{I_n} \, \forall r. \tag{1.3}$$

Molto spesso si preferisce considerare i vettori $a^{(1)}_r, a^{(2)}_r, \ldots, c^{(N)}_r$ come i vettori colonna delle matrici $A^{(1)}, A^{(2)}, \ldots A^{(N)}$ e utilizzare la notazione più compatta: $[\![A^{(1)}, A^{(2)}, \ldots, A^{(N)}]\!]$ per indicare la (1.3).

In alternativa è possibile considerare i vettori $a^{(n)}_r$ di norma unitaria e trasferire quindi la norma del r-esimo tensore di rango 1 della decomposizione, nel r-esimo elemento del vettore λ, avendo così la scrittura

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r \, a^{(1)}_r \circ a^{(2)}_r \circ \cdots \circ a^{(N)}_r = [\![\lambda; A^{(1)}, A^{(2)}, \ldots, A^{(N)}]\!].$$

Oppure si possono considerare tutti i vettori aventi la stessa norma e scrivere:

$$\mathcal{X} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r \text{ con } \|a_r\| = \|b_r\| = \|c_r\| \text{ per } r = 1 \ldots R. \tag{1.4}$$

Ovviamente questa scrittura è unica a meno del cambio di segno in $\mathbb{R}$, o della moltiplicazione per un numero della sfera unitaria in $\mathbb{C}$.

Inoltre, il rango di un tensore può variare se considerato su $\mathbb{R}$ o su $\mathbb{C}$:

Esempio. Sia $\mathcal{X} \in \mathbb{R}^{2\times 2\times 2}$:

$$\mathcal{X}(:,:,1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathcal{X}(:,:,2) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Allora le matrici che generano $\mathcal{X}$ in $\mathbb{R}$ sono:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix}.$$

Invece in $\mathbb{C}$:

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix}, \qquad B = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix}.$$

Capitolo 2

High Order SVD

La Tucker decomposition, introdotta da Tucker nel 1963, viene spesso chiamata High Order SVD, in quanto considerata una generalizzazione a più dimensioni dell'algoritmo Singular Value Decomposition delle matrici. L'obbiettivo di questa decomposizione è scrivere il tensore $\mathcal{X}$ come il prodotto mode-n di un *core tensor* $\mathcal{G}$ per una diversa matrice $A^{(n)}$ per ogni mode.

Più precisamente, per un tensore tridimensionale $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, si vuole ottenere la seguente scrittura:

$$\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, a_p \circ b_q \circ c_r. \tag{2.1}$$

Dove $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}, A \in \mathbb{R}^{I \times P}, B \in \mathbb{R}^{J \times Q}, C \in \mathbb{R}^{K \times R}$. Si possono considerare le matrici A, B, C come le componenti principali lungo i tre diversi mode e di solito sono ortogonali. Al fine di semplificare la lettura si utilizza la stessa notazione vista nella sezione precedente, $[\![\mathcal{G}; A, B, C]\!]$, per indicare la (2.1).

Elemento per elemento quindi si ha:

$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, a_p b_q c_r, \qquad \text{per i=1,...,I, j=1,...,J, k=1,...,K.}$$

Dalla relazione (1.1) possiamo scrivere in forma matriciale la decomposizione (2.1) avendo:

$$\begin{aligned} X_{(1)} &\approx A G_{(1)} (C \otimes B)^T, \\ X_{(2)} &\approx B G_{(2)} (C \otimes A)^T, \\ X_{(3)} &\approx C G_{(3)} (B \otimes A)^T. \end{aligned}$$

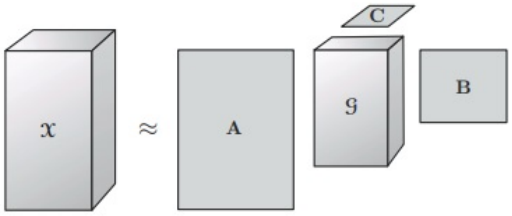

Figura 2.1: Visualizzazione della HOSVD per un tensore tridimensionale

2.1 Il rango$_n$

Si può osservare che se P, Q, R sono minori di I, J, K allora la (2.2) è effettivamente un'approssimazione e il *core tensor* $\mathcal{G}$ risulta essere una versione compressa di $\mathcal{X}$. Il vantaggio di eseguirla risiede nella memoria utilizzata, infatti conservare le informazioni contenute in $\mathcal{G}$ puó essere molto più economico in confronto a salvare l'intero tensore $\mathcal{X}$. Per contro, nel caso in cui P, Q, R siano troppo bassi, i dati contenuti in $\mathcal{G}$ possono essere significativamente diversi rispetto a quelli in $\mathcal{X}$. Per formalizzare questo concetto si introduce quindi la nozione di rango$_n$:

Definizione 2.1.1. Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_N}$ allora il rango$_n(\mathcal{X})$ è il rango dell'unfolding del tensore $\mathcal{X}$ lungo il suo mode n.

In altre parole, il rango$_n$ è il numero di colonne linearmente indipendenti di $X_{(n)}$ oppure, equivalentemente, la dimensione dello Span dei vettori mode-n di $\mathcal{X}$. Si può quindi affermare che $\mathcal{X}$ sia un tensore di rango$-(R1, R2, \ldots, R_N)$, dove ovviamente $R_n \leq I_n \, \forall n$.
Per un generico tensore $\mathcal{X}$ si può facilmente eseguire una HOSVD completa di rango $(R_1, R_2, \ldots, R_N)$, con $R_n = \text{rango}_n(\mathcal{X})$. Nel caso in cui $R_n < \text{rango}_n(\mathcal{X})$ per uno o più n, allora calcolare la decomposizione sarà più complesso e la soluzione trovata sarà ovviamente inesatta, tuttavia può rappresentare una buona inizializzazione per applicare un algoritmo iterativo.

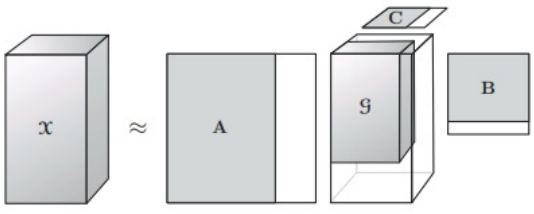

Figura 2.2: Visualizzazione della HOSVD troncata per un tensore tridimensionale

Una generalizzazione del rango$_n$ è il rango multiplo, il quale viene definito per una qualsiasi matricization del tensore, mentre il primo si riferisce esclusivamente all'unfolding trattato nella definizione 1.1.1.

2.2 L'algoritmo

L'idea dell'algoritmo è di trovare quelle componenti che riescono ad esprimere al meglio le variazioni del mode n, in modo indipendente rispetto agli altri mode. Dalla trattazione di questo algoritmo risulterà evidente perchè viene considerata la generalizzazione della SVD. Applicando questo algoritmo il *core tensor* sarà *all-orthogonal*, cioè $\sum_{i_1,i_2} s_{i_1 i_2 \alpha} s_{i_1 i_2 \beta} = \sum_{i_1,i_3} s_{i_1 \alpha i_3} s_{i_1 \beta i_3} = \sum_{i_2,i_3} s_{\alpha i_2 i_3} s_{\beta i_2 i_3} = 0$, con $\alpha \neq \beta$.

Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. Allora questo algoritmo corrisponde al seguente problema di minimo:

$$\min_{\mathcal{G},A,B,C} \|\mathcal{X} - [\![\mathcal{G}; A, B, C]\!]\|, \tag{2.2}$$

tale che $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}, A \in \mathbb{R}^{I \times P}, B \in \mathbb{R}^{J \times Q}, C \in \mathbb{R}^{K \times R}$, con A, B, C aventi le colonne ortogonali.

Riscrivendo in forma vettoriale si ottiene:

$$\|vec(\mathcal{X}) - (C \otimes B \otimes A) vec(\mathcal{G})\|.$$

La quale implica che $\mathcal{G}$ deve essere del tipo:

$$\mathcal{G} = \mathcal{X} \times_1 A \times_2 B \times_3 C.$$

Possiamo quindi riscrivere (il quadrato) della funzione che dobbiamo minimizzare:

$$\begin{aligned}
\|\mathcal{X} - [\![\mathcal{G}; A, B, C]\!]\|^2 &= \langle \mathcal{X} - [\![\mathcal{G}; A, B, C]\!], \mathcal{X} - [\![\mathcal{G}; A, B, C]\!] \rangle \\
&= \|\mathcal{X}\|^2 - 2 \langle \mathcal{X}, [\![\mathcal{G}; A, B, C]\!] \rangle + \|[\![\mathcal{G}; A, B, C]\!]\|^2 \\
&= \|\mathcal{X}\|^2 - 2 \left\langle \mathcal{X} \times_1 A^T \times_2 B^T \times_3 C^T, \mathcal{G} \right\rangle + \|\mathcal{G}\|^2 \\
&= \|\mathcal{X}\|^2 - 2 \langle \mathcal{G}, \mathcal{G} \rangle + \|\mathcal{G}\|^2 \\
&= \|\mathcal{X}\|^2 - \|\mathcal{G}\|^2 \\
&= \|\mathcal{X}\|^2 - \left\|\mathcal{X} \times_1 A^T \times_2 B^T \times_3 C^T\right\|^2.
\end{aligned}$$

Si può utilizzare un algoritmo del tipo ALS (Alternating Least Squares) per risolvere il problema: si fissano le matrici B, C e si cerca il minimo descritto. Dal momento che anche il tensore $\mathcal{X}$ è fissato, una prima soluzione è data da:

$$\max_{A} \left\|\mathcal{X} \times_1 A^T \times_2 B^T \times_3 C^T\right\|. \tag{2.3}$$

sempre con la condizione che A abbia le colonne ortogonali. La (2.3) può essere a sua volta riscritta in forma matriciale come

$$\left\|A^T W\right\| \text{ con } W = X_{(1)}(C \otimes B).$$

Si dimostra che la soluzione può essere determinata applicando l'algoritmo SVD e considerando i primi P vettori singolari sinistri di W. Successivamente, si fissano A e C e si calcola il massimo (2.3) relativo alla matrice B, e poi per C, fissando A e B. L'incremento tra due iterazioni di (2.2) converge a zero, ciò dimostra che l'algoritmo converge a una soluzione, nonostante non sia garantito che essa sia un minimo globale o un punto stazionario.

L'algoritmo per decomporre un tensore $\mathcal{X}$ di terzo ordine considerando P, Q ed R autovettori lungo i diversi mode sarà:

```
function [S,U1,U2,U3]=hosvd(X,P,Q,R)
[U1,~,~]=svds(double(tenmat(A,1,'fc')),P);
[U2,~,~]=svds(double(tenmat(A,2,'fc')),Q);
[U3,~,~]=svds(double(tenmat(A,3,'fc')),R);
S=nmodeproduct(nmodeproduct(nmodeproduct(A,U1',1),U2',2),U3',3);
```

Per avere l'approssimazione $\tilde{\mathcal{X}}$ di $\mathcal{X}$ basterà calcolare:

```
Xtilde=nmodeproduct(nmodeproduct(nmodeproduct(X,U1,1),U2,2),U3,3);
```

2.3 Assenza di unicità

La HOSVD non è unica. Si consideri ad esempio (2.1) e sia $U \in \mathbb{R}^{P\times P}$, $V \in \mathbb{R}^{Q\times Q}$, $W \in \mathbb{R}^{R\times R}$ matrici non singolari, allora

$$[\![\mathcal{G}; A, B, C]\!] = [\![\mathcal{G} \times_1 U \times_2 V \times_3 W; AU^{-1}, BV^{-1}, CW^{-1}]\!].$$

In altre parole è possibile modificare il *core tensor* moltiplicandolo con matrici non singolari conservando il grado e l'esattezza di approssimazione. Questa assenza di unicità permette di scrivere la decomposizione cercando, ad esempio, di semplificare il *core tensor* rendendolo il più sparso possibile, cercando di avere quasi tutti gli elementi uguali a zero. Un'altra scelta effettuabile è quella di rendere il valore assoluto degli elementi della super-diagonale, quindi del tipo $\mathcal{G}(i,i,i)$, il più grande possibile.

2.4 Applicazione

L'algoritmo, descritto nella sezione 2.2 viene usato in statistica e nell'analisi di dati, dove risulta molto utile quando le informazioni dipendono da tre o più indici, rendendo così naturale l'uso dei tensori.

Ho sperimentato questo algoritmo nell'ambito del riconoscimento facciale, utilizzando il database *YaleFaces* (reperibile sul sito www.face-rec.org/databases/): una collezione di immagini di 320×143 pixel in gradazioni di grigio, di $n_p = 15$ persone differenti, ognuna fotografata in $n_e = 11$ espressioni differenti. Importando e applicando un reshape, da ogni immagine si ottiene così un vettore lungo $n_i = 77760$, contenente numeri interi da 0 a 255. Salvando tutte le immagini in un unico tensore $\mathcal{X}$ si ottiene così un array di dimensione $n_i \times n_e \times n_p = 77760 \times 11 \times 15$, ovviamente si poteva optare per ottenere un tensore molto lungo rispetto invece alla seconda o alla terza componente, permutando arbitrariamente gli indici. Già con questo limitato dataset si può osservare l'enorme quantitá di memoria che occorre per salvarlo, si tratta infatti di $1,2 \cdot 10^8$ valori.

Una scelta che si può fare per occupare meno memoria è quella di operare, anzichè sul tensore completo $\mathcal{X}$, sulla sua approssimazione mediante HOSVD troncata. Come si può osservare dalla figura 2.3, questo ha un importante impatto sui dati, aggiungendo rumore e disturbando così l'immagine. Sul database considerato, ho eseguito una HOSVD troncata, prendendo solo i primi k= $60, 25, 15, 5$ autovettori sinistri del mode 1, rispetto al totale di 165 di una decomposizione completa. Lungo gli altri due mode ho considerato una SVD completa, poichè sono solamente 11 e 15 vettori, delle dimensioni relativamente basse. Di seguito ho riportato la stessa immagine del database (prima persona ritratta nella prima espressione) nelle diverse approssimazioni eseguite.

(a) k=165 (b) k=60 (c) k=25

(d) k=15 (e) k=5

Figura 2.3: $\mathcal{X}(:, 1, 1)$ al variare della quantità k di valori singolari scelti nel primo unfolding

In questo modo, ho ridotto la quantità di memoria utilizzata, passando progressivamente da $1,2 \cdot 10^8$ dati, nel caso di quella completa, fino a $3,8 \cdot 10^6$ nel caso di k= 5.

Inoltre, la quantità di tempo necessaria per eseguire l'algoritmo diminuisce, siccome si utilizzano meno valori. Tuttavia scegliendo meno autovettori, le immagini risultano essere più sfocate e indefinite, quindi i soggetti sempre più difficili da distinguere, rendendo così l'algoritmo di riconoscimento sempre meno efficace. Per verificare questo ho considerato 3 espressioni casuali per persona, creando un tensore $\mathcal{X}_{test}$ e salvando le altre 8 in $\mathcal{X}_{train}$. Successivamente ho approssimato $\mathcal{X}_{train}$ mediante HOSVD troncata con $k = 5, 10, 15, 20, 25, 120$ autovettori, dove con 120 si ha una HOSVD completa, e ho eseguito il riconoscimento delle immagini contenute in $\mathcal{X}_{test}$.

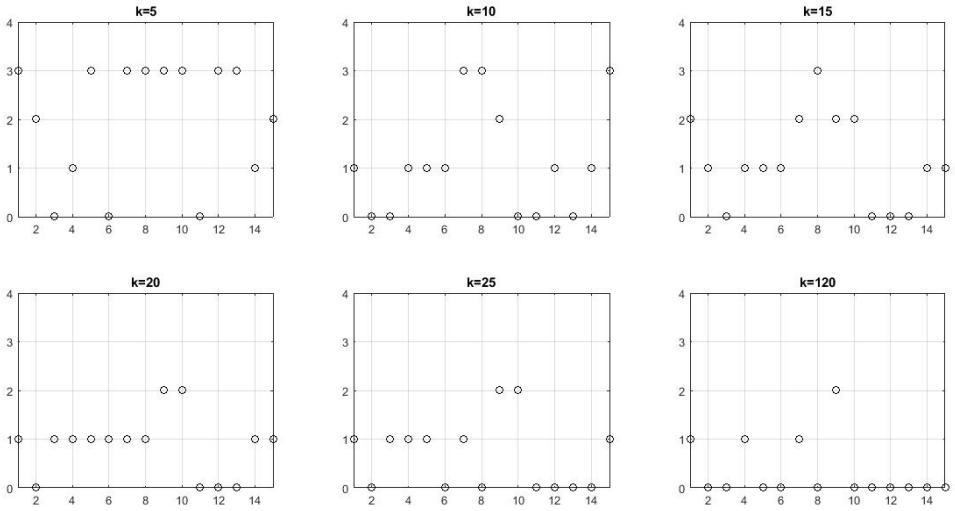

Figura 2.4: Errori del riconoscimento al variare di k autovettori

Nella figura 3.1 sono presenti le quantità di errori, cioè quante volte l'algoritmo di riconoscimento associa all'immagine della i-esima persona del tensore $\mathcal{X}_{test}$ la j-esima persona del tensore $\mathcal{X}_{train}$, con $i \neq j$. Come si può notare, considerando solo 5 autovettori, le immagini risultano essere troppo simili per poterle utilizzare; invece, con $k = 10, 15$ la percentuale di insuccesso scende circa a un terzo delle immagini totali, sebbene come abbiamo visto nella figura 2.3 le fotografie risultano quasi irriconoscibili. Mentre con $k = 20, 25$ si può già considerare una buona approssimazione in quanto la percentuale di insuccesso è pari a 20% dei test totali eseguiti, quando con un HOSVD completa questa è al 10%.

In conclusione emerge che questo algoritmo risulta molto versatile quando occorre gestire una grande quantità di dati, specialmente lavorando con molti indici, in caso contrario è più efficiente utilizzare algoritmi che sfruttano l'analisi matriciale. Inoltre si può utilizzare la versione troncata per diminuire la quantità di dati cercando di mantenere

l'accuratezza del risultato. È ovvio che il livello di approssimazione possibile dipende dal database che si sta utilizzando e dalle sue relazioni.

Capitolo 3

Canonical decomposition

La Canonical decomposition, anche chiamata PARAFAC (Parallel-Factor) e CP decomposition, fu introdotta nel 1970 da Carroll e Chang e indipendentemente da loro da Harshman.

Lo scopo di questo algoritmo è quello di scrivere un generico tensore $\mathcal{X}$ come somma di tensori di rango 1, come visto nell'equazione (1.3). Tuttavia, come già osservato, la determinazione del rango di un tensore è un problema più difficile del calcolo del rango$_n$, perchè il primo richiede lo studio del tensore come una quantità globale e non come una collezione di vettori lungo i diversi mode.

Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ si vuole scrivere

$$\mathcal{X} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r, \tag{3.1}$$

con $R \in \mathbb{R}^+$, $a_r \in \mathbb{R}^I$, $b_r \in \mathbb{R}^J$, $c_r \in \mathbb{R}^K$ per $r = 1 \dots R$.
È possibile definire $A = [a_1\, a_2 \dots\, a_R]$, allo stesso modo per le matrici B e C, inoltre si possono assumere le precedenti matrici normalizzate per colonna e considerare i pesi come un fattore moltiplicativo λ_r. Usando queste definizioni è possibile riscrivere la relazione (3.1) come:

$$X \approx [\![\lambda; A, B, C]\!] = \sum_{r=1}^{R} \lambda_r\, a_r \circ b_r \circ c_r, \tag{3.2}$$

tale che $\|a_r\| = \|b_r\| = \|c_r\| = 1$ e $\lambda_r \in \mathbb{R}$ per $r = 1 \dots R$. Riscritta in forma matriciale lungo i diversi mode, la (3.2) diventa:

$$\begin{aligned} X_{(1)} &\approx A\Lambda(C \odot B)^T, \\ X_{(2)} &\approx B\Lambda(C \odot A)^T, \\ X_{(3)} &\approx C\Lambda(B \odot A)^T, \end{aligned} \tag{3.3}$$

con ovviamente $\lambda \in \mathbb{R}^R$, $\Lambda = \mathrm{diag}(\lambda)$.

3.1 Rango tipico e rango massimo

Prima di procedere a trattare l'algoritmo vero e proprio occorre considerare alcuni aspetti intrinsechi della definizione di rango. Si è già osservato che il rango è diverso in $\mathbb{R}$ e in $\mathbb{C}$, si veda la sezione 1.4. Inoltre si dimostra che il calcolo del rango di un tensore è un problema NP-hard, si veda [2] per maggiori dettagli. Quindi dal punto di vista numerico si cerca di approssimare il tensore mediante modelli a diverso rango fissato R e considerando quello che meglio approssima il tensore di partenza (non è assicurato che questo esista). Dal punto di vista teorico vengono introdotte la nozioni di *rango massimo*, definito come il maggior rango ottenibile, e il *rango tipico*, il quale è un qualsiasi rango che si presenta con probabilità non trascurabile.

Per le matrici $I \times J$ il rango massimo e tipico coincidono e sono uguali al $\min\{I, J\}$, tuttavia per i tensori possono essere diversi. Inoltre su $\mathbb{R}$ possono esistere più valori per il rango tipico, mentre su $\mathbb{C}$ ne esiste uno solo. Ad esempio, esperimenti numerici hanno rivelato che i tensori $2 \times 2 \times 2$ con la distribuzione normale, hanno per il 79% rango 2 mentre per il 21% rango 3, da ciò si puó concludere che possono esistere tensori di rango 1 ma hanno probabilità nulla. Inoltre è noto che il rango, effettuando una permutazione dei mode, è costante.

Per un generico tensore di terzo ordine, $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, si conosce solo la seguente maggiorazione:

$$\operatorname{rank}(\mathcal{X}) \leq \min\{IJ, IK, JK\}.$$

3.2 Unicità

Una proprietà interessante dei tensori di ordine superiore a due è che la loro decomposizione è unica, mentre sappiamo che questo non vale per le matrici. Consideriamo $X = AB^T = \sum_{r=1}^{R} a_r \circ b_r$ e sia $U\Sigma V$ la SVD di X, si può scegliere $A = U\Sigma$ e $B = V$. Ma la decomposizione rimane verificata anche scegliendo $A = U\Sigma W$ e $B = VW$ con W una matrice $R \times R$ ortogonale. Infatti la SVD di una matrice è unica (sotto l'ipotesi che i valori singolari siano diversi) assumendo dei vincoli di ortogonalità, mentre la CP decomposition lo è sotto condizioni più deboli.

Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ un tensore di rango R e la sua decomposizione

$$\mathcal{X} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r = [\![A, B, C]\!].$$

Allora si può affermare che questa è l'unica possibile combinazione di tensori di rango 1, la cui somma dia $\mathcal{X}$, a meno ovviamente di permutazione o fattori di ridimensionamento. La prima eccezione si riferisce al fatto che è possibile riordinare le componenti

di rango 1 arbitrariamente: $\mathcal{X} = [\![A, B, C]\!] = [\![A\Pi, B\Pi, C\Pi]\!]$ per una qualsiasi matrice di permutazione Π. La seconda permette invece di scalare i singoli vettori:

$$\mathcal{X} = \sum_{r=1}^{R} (\alpha_r a_r) \circ (\beta_r b_r) \circ (\gamma_r c_r),$$

con $\alpha_r \beta_r \gamma_r = 1$ per $r = 1 \dots R$. Si dimostra che una condizione sufficiente per l'unicità è che:

$$\text{rank}(A) + \text{rank}(B) + \text{rank}(C) \geq 2R + 2,$$

mentre una condizione necessaria è la seguente:

$$\min\{\text{rank}(A \odot B), \text{rank}(A \odot C), \text{rank}(B \odot C)\} = R.$$

3.3 Approssimazioni di rango basso

È noto che la miglior approssimazione di rango k di una matrice è data dai suoi primi k fattori della sua SVD. Questo tipo di risultato non è vero per i tensori. Infatti è stato provato che in un tensore cubico l'approssimazione di rango 1 non è un fattore dell'approssimazione di rango 2. Un importante corollario di ciò è che le componenti di diverso rango k non possono essere calcolate sequenzialmente. Tutti i fattori quindi, a rango fissato, devono essere calcolati simultaneamente.

In generale è possibile che non esista una migliore approssimazione di rango k. In una situazione simile, risulta utile il concetto di *border rank*, definito come il minimo numero di fattori di rango 1 sufficienti per approssimare il tensore con un'approssimazione dell'errore arbitrariamente piccolo. In termini rigorosi, il border rank è definito come:

$$\widetilde{\text{rank}}(\mathcal{X}) = \min \{r \,|\, \forall \epsilon > 0, \exists \text{ un tensore } \delta\mathcal{X} \text{ t.c. } \|\delta\mathcal{X}\| < \epsilon \text{ e } \text{rank}(\mathcal{X} + \delta\mathcal{X}) = r\}.$$

Ovviamente si ha $\widetilde{\text{rank}}(\mathcal{X}) \leq \text{rank}(\mathcal{X})$.

3.4 L'algoritmo

Si supponga di cercare la migliore approssimazione di $\mathcal{X}$ a rango fissato R. Per fare ciò esistono diversi algoritmi, di seguito viene riportato quello più diffuso, che utilizza un metodo del tipo Alternating Least Squares (ALS).

Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ un tensore del terzo ordine. Si cerca di trovare una CP decomposition di R componenti fissate che meglio approssima $\mathcal{X}$, quindi si cerca:

$$\min_{\tilde{\mathcal{X}}} \left\| \mathcal{X} - \tilde{\mathcal{X}} \right\| \text{ con } \tilde{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \, a_r \circ b_r \circ c_r = [\![\lambda; A, B, C]\!]. \tag{3.4}$$

L'approccio mediante ALS consiste nel fissare due matrici per risolvere il minimo (3.4) per la terza matrice, ripetendo questa procedura alternativamente finchè non viene verificato il criterio di convergenza.

Si supponga, per esempio, di aver fissato B e C, ricordando le relazioni (3.3) per i diversi mode, il minimo che si sta cercando in (3.4) diventa

$$\min_{\tilde{A}} \left\| X_{(1)} - \tilde{A}(C \odot B)^T \right\|_F . \tag{3.5}$$

Con $\tilde{A} = A \cdot \operatorname{diag} \Lambda$. Questo problema di minimi quadrati può essere risolto mediante l'utilizzo della matrice pseudoinversa di Moore-Penrose.

$$\tilde{A}^T = \left[(C \odot B)^T (C \odot B) \right]^{-1} (C \odot B)^T.$$

Ma per definizione: $\left[(C \odot B)^T (C \odot B) \right]^{-1} (C \odot B)^T =: (C \odot B)^\dagger$, sostituendo si ha quindi:

$$\tilde{A} = X_{(1)}((C \odot B)^\dagger)^T = X_{(1)}((C \odot B)^T)^\dagger. \tag{3.6}$$

Ricordando la proprietà (1.2) si può riscrivere la matrice:

$$\tilde{A} = X_{(1)}(C \odot B)(C^T C * B^T B)^\dagger. \tag{3.7}$$

In questo modo occorre calcolare la pseudoinversa di una matrice $R \times R$ piuttosto che di una $JK \times R$. D'altro canto, questa versione non è sempre consigliata a causa del possibile mal condizionamento della matrice. Mediante la (3.7) è possibile calcolare le colonne di A ponendo $\lambda_r = \|\tilde{a}_r\|$ e quindi si ha $a_r = \tilde{a}_r / \lambda_r$.

Successivamente si ripete questo procedimento fissando A e C per risolvere (3.5) per la matrice B e infine per C, considerando A e B fissate.

Il metodo ALS è semplice e facile da implementare, tuttavia può richiedere diverse iterazioni per convergere. Inoltre non è garantita la convergenza a un minimo globale o a un punto stazionario, ma soltanto a una soluzione nella quale la funzione oggetto di (3.4) cessa di incrementare. La soluzione finale può dipendere profondamente dal punto di partenza.

La parte centrale per la prima microiterazione implementata risulterà essere:

```
function [A,B,C]=cp_als(X,R)
...
B_hat=B'*B;
C_hat=C'*C;

for i=1:maxit
    V=C_hat*B_hat;
    Atilde=double(tenmat(X,1,'bc'))*(khatrirao(C,B))*pinv(V);
```

```
A=Atilde/diag(normacol(Atilde));      %normalizza le colonne di Atilde

Xnew=rankcomp(Atilde,B,C);      %crea la approssimazione X^(k+1)
normares=norm(Xnew-X);
normait=norm(Xnew-Xold);
if (normait<tol)
    break
end
A_hat=A'*A;
```

...

Ovviamente questa porzione di codice va ripetuta per le matrici B e C. L'inizializzazione può essere fatta in modo random oppure considerando i primi R autovettori dei diversi mode, se R è abbastanza piccolo. Come si può osservare l'implementazione risulta più elaborata della HOSVD, inoltre, essendo un metodo iterativo, richiede più parametri come il valore `tol` della tolleranza desiderata.

3.5 Convergenza locale

Di seguito si riportano alcuni risultati teorici di convergenza locale. La difficoltà di questi risultati risiede nella possibilità di scalare i fattori, mentre l'assenza di unicità per permutazioni non ha effetto localmente.

Siano $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ e $R \in \mathbb{N}$ il rango fissato, sia $\mathbf{X} = \mathbb{R}^{I\times R} \times \mathbb{R}^{J\times R} \times \mathbb{R}^{K\times R}$ e si consideri la funzione:

$$f : \mathbf{X} \to \mathbb{R}; \quad x = (A, B, C) \mapsto \frac{1}{2}\left\|\mathcal{X} - \sum_{r=1}^{R} a_r \circ b_r \circ c_r\right\|^2 = \frac{1}{2}\left\|\mathcal{X} - \tau(A, B, C)\right\|^2. \quad (3.8)$$

Posto $\tau : \mathbf{X} \to \mathbb{R}^{I\times J\times K}; \quad x = (A, B, C) \mapsto \sum_{r=1}^{R} a_r \circ b_r \circ c_r$, si sta quindi cercando:

$$\min_{x\in\mathbf{X}} f(x) = f(A, B, C). \quad (3.9)$$

Si supponga che esista un minimo locale che sarà indicato con $x^* = (A^*, B^*, C^*)$.

Come si è visto nella sezione precedente ad ogni iterazione il metodo calcola il punto $(n+1)$-esimo delle seguenti successioni:

$$\begin{aligned} A^{(n+1)} &= \arg\min_{A\in\mathbb{R}^{I\times R}} f(A, B^{(n)}, C^{(n)}), \\ B^{(n+1)} &= \arg\min_{B\in\mathbb{R}^{J\times R}} f(A^{(n)}, B, C^{(n)}), \\ C^{(n+1)} &= \arg\min_{C\in\mathbb{R}^{K\times R}} f(A^{(n)}, B^{(n)}, C); \end{aligned} \quad (3.10)$$

Questo algoritmo è un esempio particolare del metodo di Gauss-Seidel (rilassato) per blocchi non-lineari. Si è dimostrato che ogni microiterazione possiede un'unica soluzione applicando la pseudoinversa, allora a ogni iterazione si definisce un operatore S:

$$(A^{(n+1)}, B^{(n+1)}, C^{(n+1)}) = x^{(n+1)} = S(x^{(n)}) = S(A^{(n)}, B^{(n)}, C^{(n)}). \tag{3.11}$$

D'ora in poi si considerino solo i punti di minimo locali nell'aperto

$$\hat{\mathbf{X}} = \{(A, B, C) \in \mathbf{X} |\, a_r \neq 0,\, b_r \neq 0,\, c_r \neq 0 \text{ per } r = 1 \dots R\}\,.$$

Si assuma che un tale minimo esista. Supponendo $x^* \in \hat{\mathbf{X}}$, si può pensare che (3.10) abbia un'unica soluzione. È lecito inoltre supporre che i minimi locali in $\mathbf{X} \backslash \hat{\mathbf{X}}$ siano troppo degeneri per le applicazioni dell'algoritmo, infatti per i tensori appartenenti a questo spazio almeno una componenete di rango 1 è nulla, rendendo necessario modificare il parametro R.

Dalla sezione 3.2 si può dedurre che ogni termine $a_r^* \circ b_r^* \circ c_r^*$ può essere sostituito con $(\alpha_r a_r) \circ (\beta_r b_r) \circ (\gamma_r c_r)$, e quest'ultimo è sempre un minimo locale di f, e quindi un punto fisso di S, finchè $\alpha_r \beta_r \gamma_r = 1$. Quindi non è possibile trovare dei minimi isolati. In più, è possibile che un termine tenda all'infinito mentre un altro tenda a zero, ad esempio $a_1^{(n)}$ e $b_1^{(n)}$, questo comporta che il loro prodotto resti comunque limitato, ma peggiora il condizionamento della matrice ad ogni microiterazione.

Per queste ragioni si preferisce applicare una normalizzazione, come già visto nella sezione 1.4, con la quale si ha una scrittura del tipo (1.4).

La normalizzazione di un tensore in questa forma, senza cambiare il segno ai vettori, definisce un operatore $R(A, B, C)$ definito sui vettori colonna delle matrici:

$$(a_r, b_r, c_r) \mapsto \left(\frac{\delta_r a_r}{\|a_r\|}, \frac{\delta_r b_r}{\|b_r\|}, \frac{\delta_r c_r}{\|c_r\|} \right), \quad \delta_r = (\|a_r\|\, \|b_r\|\, \|c_r\|)^{\frac{1}{3}}, \text{ per } r = 1 \dots R.$$

Da notare che R può essere definito su tutto $\mathbf{X}$ come estensione continua (su $\mathbf{X} \backslash \hat{\mathbf{X}}$ viene posto 0), ma è differenziabile solo su $\hat{\mathbf{X}}$.

Per i punti fissi dell'algoritmo vale necessariamente $R(A^*, B^*, C^*) = (A^*, B^*, C^*)$ e per questo motivo si dicono *equilibrati*.

3.5.1 Teorema di convergenza

Sempre per l'assenza di unicità si deduce che f è costante nella sotto-varietà $2R$-dimensionale (nel caso reale non connessa):

$$\mathcal{M}^* = \left\{ (A^* \Delta_1, B^* \Delta_2, C^* \Delta_1^{-1} \Delta_2^{-1}) \in \mathbf{X} |\, \Delta_1, \Delta_2 \text{ matrici diagonali non singolari} \right\},$$

che contiene tutte le possibili rappresentazioni di x^*, che si suppone essere in $\hat{\mathbf{X}}$, ottenute mediante ridimensionamento dei vettori a_r^*, b_r^*, c_r^* (anche cambiandoli di segno).

Ogni punto di $\mathcal{M}^*$ è minimo locale di (3.9), di conseguenza la derivata f' è nulla su tutto $\mathcal{M}^*$ e quindi l'Hessiana di $f''(x^*)$ ha al più rango $\dim \mathbf{X} - 2R = R(I+J+K) - 2R$. Più precisamente sia $T\mathcal{M}^*_{z^*}$ lo spazio tangente a $\mathcal{M}^*$ in x^*. Allora $\langle h, f''(x^*)h\rangle = 0$ per ogni $h \in T\mathcal{M}^*_{x^*}$, e si può osservare che:

$$T\mathcal{M}^*_{x^*} = \{(A^*\Delta_1, B^*\Delta_2, -C^*(\Delta_1+\Delta_2)) \in \mathbf{X} |\, \Delta_1, \Delta_2 \text{ matrici diagonali}\}\,.$$

Ipotesi 1. Il rango di $f''(x^*)$ è esattamente $R(I+J+K)-2R$, cioè $\ker(f''(x^*)) = T\mathcal{M}^*_{x^*}$.

In altre parole, con l'ipotesi 1 si sta assumendo che $f''(x^*)$ sia definita positiva in tutte le direzioni eccetto quelle tangenti al ridimensionamento. Questo implica che la parametrizzazione di x^* è localmente unica.

Lemma 3.5.1. *$R'(x^*)$ è un proiettore e* $\ker(R'(x^*)) = T\mathcal{M}^*_{x^*}$.

Dimostrazione. Si dimostra solo nel caso reale.

Si osserva anzitutto che $R = R \circ R$ implica che $R'(x^*)$ è un proiettore:

$$R'(x^*) = R'(R(x^*))R'(x^*) = R'(x^*)R'(x^*).$$

Inoltre R è costante sulle componenti connesse di $\mathcal{M}^*$ e quindi $R'(x^*)h = 0$ per ogni $h \in T\mathcal{M}^*_{x^*}$.

Come già osservato R è l'identità sull'insieme degli x equilibrati, in particolare sulla sottovarietà degli (A, B, C) le cui colonne hanno la stessa norma delle colonne di (A^*, B^*, C^*). Quindi $R'(x^*)h = h$ per h che appartiene allo spazio tangente di questa sottovarietà. Poichè quest'ultimo può essere considerato come un prodotto di sfere è chiaro che il suo spazio tangente sia

$$U = \{(A, B, C) \in \mathbf{X} |\, a_r \perp a^*_r,\ b_r \perp b^*_r,\ c_r \perp c^*_r \text{ per } r = 1 \dots R\}\,.$$

Infine si può osservare che $R'(x^*)h \neq 0$ per tutti $h \in V \setminus \{0\}$ con

$$V = \{(A^*\Delta, 0, 0) \in \mathbf{X} |\, \Delta \text{ matrice diagonale}\}\,.$$

Questo termina la dimostrazione e in più, si è provato che: $\dim T\mathcal{M}^*_{x^*} + \dim(U \oplus V) = \dim \mathbf{X}$. □

Lemma 3.5.2. *Assumendo l'ipotesi 1 allora l'operatore ALS S in (3.11) è ben definito e differenziabile in un intorno di x^* che è un punto fisso di S.*
Inoltre $R \circ S$ è localmente ben definita e differenziabile.

Di conseguenza è possibile effettuare un'iterazione dell'algoritmo.

Dimostrazione. Si dimostra che i blocchi diagonali di $f''(x^*)$ sono definiti positivi: si consideri $h = (H_A, 0, 0)$ con $H_A \in \mathbb{R}^{I\times R}$, $H_A \neq 0$. Dal momento che $h \notin T\mathcal{M}^*_{x^*}$, l'ipotesi 1 garantisce che $\langle h, f''(x^*)h\rangle > 0$, questo implica che il primo blocco è definito positivo; lo stesso procedimento si può reiterare per il secondo e il terzo blocco.

Da ciò segue che la diagonale di $f''(x)$ è definita positiva per tutti gli x abbastanza vicini a x^*. Ogni microiterazione del tipo (3.10) è un problema ai minimi quadrati, e la matrice associata al sistema è il corrispondente blocco diagonale dell'Hessiana $f''(x)$. Quindi ogni microiterazione possiede un'unica soluzione globale e chiaramente x^* è punto fisso di S. Quindi è possibile scegliere un intorno abbastanza piccolo da poter eseguire tutte le microiterazioni consecutivamente con un'unica soluzione, cioè S è ben definita e differenziabile localmente.

Dal momento che x^* è in $\hat{\mathbf{X}}$ allora $S(x^{(n)}) \in \hat{\mathbf{X}}$, che è aperto, se $x^{(n)}$ è abbastanza vicino a x^*, ovvero $(R \circ S)(x^{(n)})$ è ben definita e differenziabile. □

Se si assume l'ipotesi 1 allora per il lemma 3.5.1

$$|x|_*^2 = \|(I - R'(x^*)x\|^2 + |x|_E^2$$

definisce una norma su $\mathbf{X}$, con $|x|_E^2$ che denota la seminorma energia di $f''(x^*)$, cioè: $|x|_E^2 = \langle x, f''(x^*)x\rangle^{\frac{1}{2}}$.

Teorema 3.5.1. *Sia $x^* = (A^*, B^*, C^*)$ un punto equilibrato di minimo locale di f tale che valga l'ipotesi 1. Allora per ogni $\epsilon > 0$ esiste un intorno di x^* tale che per un punto iniziale $x^{(0)}$ in questo intorno, le iterazioni dell'algoritmo convergono linearmente a x^*, e in particolare soddisfano*

$$\left|x^{(n+1)} - x^*\right|_* \leq (q + \epsilon)\left|x^{(n)} - x^*\right|_*,$$

dove $q = |S'(x^*)| < 1$.

La sequenza di tensori $(\tau(x^{(n)}))$ converge R-linearmente a $\tau(x^*)$ con la stessa velocità asintotica:

$$\limsup_{n\to\infty} \left\|\tau(x^{(n)}) - \tau(x^*)\right\|^{\frac{1}{n}} \leq q.$$

Con $\tau : \boldsymbol{X} \to \mathbb{R}^{I\times J\times K} \quad x = (A, B, C) \mapsto \sum_{r=1}^{R} a_r \circ b_r \circ c_r$.

Quindi posto x^* un minimo locale equilibrato per f tale che valga l'ipotesi 1 allora l'iterazione $x^{(n+1)} = (R \circ S)(x^{(n)})$ localmente è linearmente convergente.

Dimostrazione. Dal lemma 3.5.2 sappiamo che $(R \circ S)$ è ben definito e differenziabile e x^* è un suo punto fisso. Si deve solo provare che $|(R \circ S)'(x^*)|_* \leq q < 1$. È vero che $f(R(x^* + h)) = f(x^* + h)$ per un h sufficientemente piccolo. Dal momento che $f'(x^*) = f'(R(x^*))$ segue che

$$|R'(x^*)h|_E^2 = \langle R'(x^*)h, f''(x^*)R'(x^*)h\rangle = \langle h, f''(x^*)h\rangle = |h|_E^2.$$

In più dal lemma 3.5.1 $(I - R'(x^*))R'(x^*) = 0$. Quindi per tutti gli $h \in \mathbf{X}$ si ha:

$$|(R \circ S)'(x^*)h|_* = |R'(x^*)S'(x^*)h|_* = |S'(x^*)h|_E \leq |S'(x^*)|_E \, |h|_E \leq |S'(x^*)|_E \, |h|_* .$$

Dal lemma 3.5.2, $S'(x^*)$ è la matrice di iterazione degli errori del metodo lineare a blocchi di Gauss-Seidel per $f''(x^*)$. È noto che quest'ultima sia una contrazione nella seminorma energia, cioè $|S'(x^*)|_E < 1$ se $f''(x^*)$ è semidefinita positiva con un blocco definito positivo. Quest'ultima condizione è assicurata dall'ipotesi 1. Siccome τ è lipshitziana e continua su un sottoinsieme compatto di $\mathbf{X}$, ne segue che

$$\limsup_{n\to\infty} \left\|\tau(x^{(n)}) - \tau(x^*)\right\|^{\frac{1}{n}} \leq \limsup_{n\to\infty} \left|x^{(n)} - x^*\right|_*^{\frac{1}{n}} \leq q.$$

Si può provare che $q = |S'(x^*)|_E$, concludedendo quindi la dimostrazione della prima parte. Per la dimostrazione della seconda parte si utilizza il teorema di Banach-Caccioppoli. Infatti quest'ultimo assicura che l'algoritmo è linearmente convergente in un intorno del suo punto fissato $x^* \in \hat{\mathbf{X}}$ se il raggio spettrale di $(R \circ S)'(x^*) = R'(x^*)S'(x^*)$ è minore di 1, così che $(R'(x^*)S'(x^*))^n \to 0$ per $n \to \infty$. Siccome $S'(x^*)$ è l'identità sullo spazio nullo $T\mathcal{M}^*_{x^*}$ di $R'(x^*)$ questo è equivalente a $R'(x^*)(S'(x^*))^n \to 0$ per $n \to \infty$. Da quanto visto in precedenza sul ruolo di $S'(x^*)$ allora si può concludere che per ogni $h \in \mathbf{X}$, la successione $S'(x^*)^n h$ converge a un elemento del ker di $f''(x^*)$ (con la condizione che $f''(x^*)$ sia semidefinita positiva con un blocco diagonale definito positivo). Ciò conclude la dimostrazione. □

Si può osservare che questo teorema è valido considerando una qualsiasi funzione costo $J : \mathbb{R}^{I\times J\times K} \to \mathbb{R}$, considerando quindi $f(A, B, C) = J(\tau(A, B, C))$. Ad esempio, una possibile scelta per J potrebbe essere la norma energia per operatori differenziali parziali autoaggiunti. Nel caso generale in cui J non sia lineare allora le microiterazioni in (3.10) non possiedono necessariamente una soluzione unica nonostante l'ipotesi sulla matrice Hessiana. Una soluzione potrebbe essere sostituire in ogni microstep l'espansione al secondo ordine di J e minimizzarla.

3.5.2 Osservazioni sull'ipotesi 1

Siano τ e f definiti come in (3.8), allora

$$\langle h, f''(x^*)h\rangle = \|\tau'(x^*)h\|^2 + \langle h, (\tau(x^*) - \mathcal{X}, \tau''(x^*)h\rangle . \tag{3.12}$$

L'ipotesi 1 afferma che $\langle h, f''(x^*)h\rangle = 0$ solo se $h \in T\mathcal{M}^*_{x^*}$. Quindi necessariamente per tali h si ha $\tau'(x^*)h = 0$ (siccome τ è costante su $\mathcal{M}$) e quindi anche $\langle h, \tau(x^*) - \mathcal{X}, \tau''(x^*))h\rangle = 0$.

Esempio. Si vuole fornire l'esempio di un tensore di $R = 3$ per cui non vale l'ipotesi 1.

Sia $\mathcal{T}$ tale che $\mathcal{T}_{i_1 i_2 i_3} = \sin(i_1 + i_2 + i_3)$. Si puó provare che

$$\sin(i_1 + i_2 + i_3) = \sum_{r=1}^{3} \sin(i_r + \beta_r) \prod_{\substack{k=1 \\ k \neq r}} \frac{\sin(i_r + \beta_r + \alpha_k - \alpha_r)}{\sin(\alpha_k - \alpha_r)},$$

per tutti $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}$ con $\sin(\alpha_k - \alpha_r) \neq 0$ per $k \neq r$ e per tutti $\beta_1, \beta_2, \beta_3 \in \mathbb{R}$ con $\beta_1 + \beta_2 + \beta_3 = 0$. Quindi se $\tau'(x^*) = \mathcal{T}$ l'esatta decomposizione data in precedenza, si può scalare in modo differenziabile i fattori. Geometricamente significa che τ e f sono costanti su una sottovarietà di minimi globali di dimensione maggiore di $2R$. Di conseguenza $\ker(\tau'(x^*))$ e $\ker(f''(x^*))$ sono più estesi di $T\mathcal{M}^*_{x^*}$ e non vale quindi l'ipotesi 1

In alternativa all'ipotesi 1 si può quindi assumere:

Ipotesi 2. $\tau'(x^*)h \neq 0$ per tutti $h \notin T\mathcal{M}^*_{x^*}$, cioè $\ker(\tau'(x^*)) = T\mathcal{M}^*_{x^*}$.

In questo modo si suppone che la rappresentazione di $\tau(x^*)$ sia essenzialmente unica, escludendo i casi come l'esempio precedente. Nel caso in cui $\tau(x^*) = \mathcal{X}$ è una decomposizione esatta, allora da (3.12) si osserva che 1 è equivalente a 2 e vale:

Teorema 3.5.2. *Se $\tau(x^*) = \mathcal{X}$ e vale l'ipotesi 1, allora in un intorno di x^* l'algoritmo ALS è linearmente convergente a x^*.*

3.6 Applicazione

Come si è già visto calcolare il rango di un tensore è un problema NP-hard. Supponiamo che si voglia calcolare il rango del tensore $\mathcal{X}$, contenente tutte le immagini del YaleFaces database. Allora una soluzione potrebbe essere approssimarlo mediante una canonical decomposition di rango R, facendo variare R in un dato intervallo di interi. Successivamente si può considerare R tale che verifichi $\min \|\mathcal{X} - \mathcal{X}_R\|$ e prendere quindi $\mathcal{X}_R$ come la decomposizione di $\mathcal{X}$. È ovvio che questa operazione richiede un elevato numero di calcoli per poter essere eseguita, quindi è consigliabile solo se $\mathcal{X}$ è di ridotte dimensioni

Il grafico mostra la norma del residuo della decomposizione al variare di $R = 1 \dots 30$. Si osserva anzi tutto che il minimo si ottiene per $R = 12$ ed è dell'ordine di 10^5, nel caso in cui si ritenga questo valore troppo elevato, si può eseguire la stessa procedura in un secondo intervallo, trovando così un altro valore R' che realizza il minimo e verificare per quale valore tra R e R' verifica $\min\{\|\mathcal{X} - \mathcal{X}_R\|, \|\mathcal{X} - \mathcal{X}'_R\|\}$, reiterando una terza volta fino a un R soddisfacente.

Notiamo che il grafico non segue un andamento monotono, ma sono presenti molte oscillazioni, questo perchè non vi sono relazioni tra le approssimazioni di rango diverso. Quindi non è assicurato che a una buona approssimazione di rango R, ne corrispondi una migliore di rango $R + 1$.

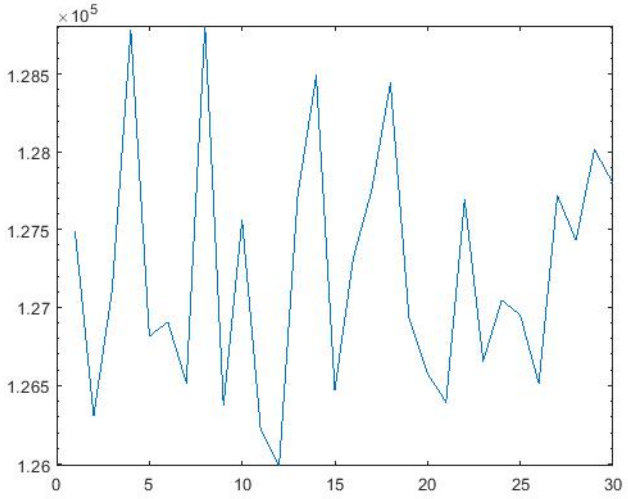

Figura 3.1: $\|\mathcal{X} - \mathcal{X}_R\|$ con $R = 1 \dots 30$

Esempio. Vogliamo esporre un esempio di canonical decomposition per un tensore $\mathcal{X} \in \mathbb{R}^{4\times3\times2}$, di piccole dimensioni. In questo caso risulta più comodo partire proprio dalle sue componenti di rango 1 che lo generano.

Siano date le matrici:

$$A = \begin{bmatrix} 5 & 3 \\ 25 & 14 \\ 17 & 4 \\ 30 & 29 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 27 \\ 24 & 3 \\ 25 & 12 \end{bmatrix} \qquad C = \begin{bmatrix} 8 & 13 \\ 25 & 28 \end{bmatrix}$$

L'unfolding lungo il mode 1 dei due tensori ottenuti come prodotto esterno dei vettori colonna sono:

$$\mathcal{U}_{(1)} = \left[\begin{array}{ccc|ccc} 40 & 960 & 1000 & 125 & 3000 & 3125 \\ 200 & 4800 & 5000 & 625 & 15000 & 15625 \\ 136 & 3264 & 3400 & 425 & 10200 & 10625 \\ 240 & 5760 & 6000 & 750 & 18000 & 18750 \end{array}\right],$$

$$\mathcal{V}_{(1)} = \left[\begin{array}{ccc|ccc} 1053 & 117 & 468 & 2268 & 252 & 1008 \\ 4914 & 546 & 2184 & 10584 & 1176 & 4704 \\ 1404 & 156 & 624 & 3024 & 336 & 1344 \\ 10179 & 1131 & 4524 & 21924 & 2436 & 9744 \end{array}\right].$$

La cui somma sarà il tensore $\mathcal{X} = [\![A, B, C]\!]$ così fatto:

$$\mathcal{Z}_{(1)} = \left[\begin{array}{ccc|ccc} 1093 & 1077 & 1468 & 2393 & 3252 & 4133 \\ 5114 & 5346 & 7184 & 11209 & 16176 & 20329 \\ 1540 & 3420 & 4024 & 3449 & 10536 & 11969 \\ 10419 & 6891 & 10524 & 22674 & 20436 & 28494 \end{array}\right].$$

Siamo interessati a calcolare le sue 2 componenti di rango 1, che saranno a priori diverse dalle matrici A, B, C. Ovviamente l'algoritmo, come lo abbiamo presentato, restituirà 3 matrici $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ aventi colonne normalizzate e un vettore λ contenente le norme dei due tensori di rango 1:

$$\tilde{A} = \begin{bmatrix} 50.1241 & -0.1028 \\ 0.6626 & -0.5849 \\ 0.6821 & -0.7882 \\ 0.2834 & 0.1611 \end{bmatrix}, \qquad \tilde{B} = \begin{bmatrix} 0.5337 & 0.6096 \\ 0.4954 & 0.4434 \\ 0.6854 & 0.6571 \end{bmatrix}, \qquad \tilde{C} = \begin{bmatrix} 0.3559 & 0.3623 \\ 0.9345 & 0.9321 \end{bmatrix},$$

$$\lambda = \begin{bmatrix} 113710.5413 & 78282.59599 \end{bmatrix}.$$

Calcolando $\tilde{\mathcal{Z}} = [\![\lambda; \tilde{A}, \tilde{B}, \tilde{C}]\!]$ si può verificare che $\left\|\mathcal{Z} - \tilde{\mathcal{Z}}\right\| = 546.4997$, mentre $\frac{\left\|\mathcal{Z}-\tilde{\mathcal{Z}}\right\|}{\|\mathcal{X}\|} = 0.0096$, quindi si può considerare già una buona approssimazione.

Bisogna osservare che, inizializzando l'algoritmo a valori casuali, è molto improbabile che eseguendolo una seconda volta otteniamo lo stesso risultato, invece se scegliamo di inizializzarlo ai primi R autovalori del tensore (se le dimensioni lo permettono) questa probabilità aumenta. Inoltre non è assicurato che $\left\|\mathcal{Z} - \tilde{\mathcal{Z}}\right\|$ sia zero, infatti come abbiamo visto si ha convergenza solo per $\left\|\mathcal{Z}^{(k+1)} - \mathcal{Z}^{(k)}\right\|$, ciò non assicura che il residuo sia nullo. Va sottolineato che, in generale, partendo dalla somma di n tensori di rango 1, non si ottiene un tensore di rango n, ma si può solo concludere che $\text{rank}(\mathcal{Z}) \leq n$, siccome a priori non è garantito che n sia il minimo della definizione 1.4.1.

Conclusioni

In questo lavoro abbiamo voluto dare una panoramica di due delle decomposizioni tensoriali più utilizzate: la High Order Singular Value decomposition e la Canonical decomposition. Come abbiamo potuto osservare sono entrambe facili da implementare e sono numerosi gli ambiti in cui queste trovano applicazioni: problemi di elaborazioni di segnali, i big data e data mining, le PDE, la psicometria, la chemometria giungendo fino alla fisica matematica.

In particolare, in questa tesi abbiamo studiato l'utilizzo dei tensori nel riconoscimento facciale, soffermandoci sulle due decomposizioni più diffuse: l'HOSVD e la canonical decomposition. Abbiamo analizzato l'analogia della prima con la SVD matriciale e quindi il suo naturale utilizzo per approssimare, ad esempio ai primi k autovettori, un database contenente immagini di persone. Questo metodo risulta essere molto semplice da implementare e non è iterativo, per cui si è certi dell'esattezza del risultato, tuttavia, come abbiamo osservato, utilizzando un valore k troppo basso, si aumenta il rumore e la distorsione dei dati.

Successivamente abbiamo analizzato la canonical decomposition, in particolare del problema del calcolo del rango, che risulta essere NP-hard. Per questo, è sconsigliato calcolare il rango di un tensore molto esteso, e preferire quindi un approccio che utilizza l'algoritmo HOSVD in questo caso. Abbiamo inoltre riportato risultati di convergenza locale, che quindi implicano che eseguendo sullo stesso tensore la decomposizione non si otterranno gli stessi risultati. Inoltre, essendo un metodo iterativo occorre un'izializzazione (che può essere sia casuale sia ricavata mediante HOSVD, considerando i primi R autovettori). Questo metodo risulta essere molto utile quando si è in presenza di vettori di cui si conosce già in precedenza il rango tipico e il rango massimo e quindi si è certi che il metodo convergerà abbastanza velocemente se fissato il giusto rango R.

Bibliografia

[1] Lieven De Lathauwer, Bart De Moor and Joos Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (4), 2000

[2] J. Hȧstad, *Tensor rank is NP-complete*, J.Algorithms, 11 (1990)

[3] T. Kolda and B. Bader,*Tensor Decompositions and Applications*, SIAM Review, 51 (3), 2009

[4] André Ushmajew, *Local convergence of alternating least squares algorithm for canonical tensor approximation*, SIAM J. Matrix Anal. Appl., 33 (2), 2012