Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

Decomposizioni tensoriali

e metodi per il riconoscimento di Immagini

Tesi di Laurea in Analisi Numerica

Relatore:
Chiar.ma Prof.ssa
Valeria Simoncini

Presentata da:
Alessandro Gardenghi

Anno Accademico 2023/2024

ai miei amici

Introduzione

Negli ultimi anni l'analisi di enormi quantità di dati è diventato un problema sempre più ricorrente in innumerevoli e diversissimi ambiti, come la statistica, il Machine Learning, la finanza, il pattern recognition. La ricerca di strumenti e metodi ottimali per lavorare con Dataset di grandi dimensioni risulta quindi essere fondamentale. L'oggetto matematico che meglio si presta a questo tipo di analisi sono i tensori, che possono essere immaginati come la generalizzazione a più dimensioni di una tabella. Queste strutture, grazie alla loro multidimensionalità, riescono a rappresentare dati dipendenti da molte variabili, in maniera ordinata e di più facile interpretazione.

Lo scopo di questa tesi è quello di presentare i tensori sia come oggetto matematico astratto che come struttura concreta per raccogliere e analizzare dati.

Nel primo Capitolo dell'elaborato vengono introdotti i tensori dal punto di vista teorico, e viene data particolare rilevanza ad alcune operazioni fondamentali come il *mode-n* unfolding e il prodotto modale, oltre che alla definizione di tensore di rango uno.

I Capitoli 2 e 3 trattano le due più importanti decomposizioni tensoriali, rispettivamente la decomposizione canonica (detta CP) e l'HOSVD. La prima si basa sulla definizione di tensore di rango uno ed è strettamente collegata alla nozione di rango di un tensore (anch'essa esposta nel Capitolo 2). La seconda è considerata la generalizzazione della SVD matriciale ai tensori. In entrambi i capitoli vengono analizzati diversi aspetti delle due decomposizioni, dall'unicità, all'utilizzo di esse per approssimare un tensore, fino ad alcuni aspetti algoritmici, quali l'implementazione, il costo computazionale, ...

Nell'ultimo Capitolo viene presentato il riconoscimento di immagini, un'applicazione classica nel Data Science, ed in particolare ci concentreremo sul problema del riconoscimento di caratteristiche. Più precisamente viene analizzato il riconoscimento da parte della macchina di immagini di cifre scritte a mano tramite differenti algoritmi. Il Dataset utilizzato per valutare i diversi metodi è MNIST. Viene trattato con particolare attenzione un algoritmo basato sull'HOSVD, un esempio di applicazione dei tensori nel Data Mining. La tesi si conclude con un'analisi tecnica dell'algoritmo, al fine di cercare i parametri di inizializzazione che rendano l'HOSVD maggiormente efficiente.

Notazione

In tutti i capitoli della tesi i vari oggetti matematici sono indicati con:

- carattere calligrafico per i tensori, $\mathcal{X}$;
- grossetto maiuscolo per le matrici, $\boldsymbol{A}$;
- grossetto minuscolo per i vettori, $\boldsymbol{a}$;
- corsivo per gli scalari, α.

Indice

Capitolo 1

I Tensori

In questo capitolo viene introdotta la nozione di tensore ed alcune definizioni e proprietà ad essa correlate; i testi di riferimento sono l'articolo *Tensor Decompositions and Applications* di T. Kolda e B. Bader [1] e la Sezione 12.4 del libro *Matrix Computations*, 4 Ed., di G. Golub e Ch. Van Loan [3].

Un tensore è una tabella multidimensionale, il cui numero di dimensioni è detto ordine o modo. Più formalmente un tensore $\mathcal{X}$ di ordine N è un elemento del prodotto di N $\mathbb{R}$-spazi vettoriali (o $\mathbb{C}$-spazi vettoriali): $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Esempi elementari di tensori sono i vettori e le matrici, che corrispondono rispettivamente ai tensori di primo e secondo ordine.

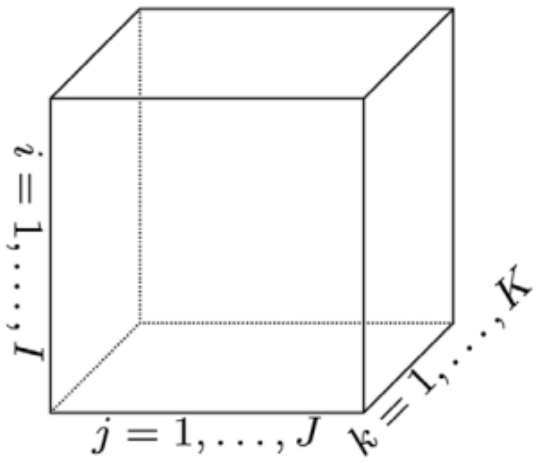

Figura 1.1: *Una possibile rappresentazione di* $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

L'elemento di $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in posizione $i_1, \ldots, i_N$, $\mathcal{X}(i_1, \ldots, i_N)$ con $1 \leq i_j \leq I_j$, viene indicato come $x_{i_1,\ldots,i_N}$; mentre con il termine *k-esima* fibra, o *mode-k* fibra, si

intende un vettore appartenente a $\mathbb{R}^{I_k}$ ottenuto fissando tutte le coordinate di $\mathcal{X}$ tranne la *k-esima*.

È possibile definire un prodotto interno sullo spazio dei tensori:

Definizione 1.1 (**Prodotto interno**). *Siano $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ allora il prodotto interno tra due tensori è definito come*

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

Nel caso in cui il prodotto interno di due tensori sia uguale a zero essi si dicono ortogonali.

Di conseguenza è immediato definire una norma sullo spazio dei tensori:

Definizione 1.2 (**Norma tensoriale**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ allora la norma di $\mathcal{X}$ è definita come*

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

Si osserva che questa norma corrisponde alla norma di Frobenius di $\mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$.

Definizione 1.3 (**Tensore diagonale**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore, si dice diagonale se $x_{i_1,i_2,\ldots,i_N} \neq 0$ solo se $i_1 = i_2 = \cdots = i_N$; per $i_k = 1, \ldots, I_k$ e $k = 1, \ldots, N$.*

1.1 Unfolding

L'unfolding o matricizzazione di un tensore $\mathcal{X}$ di ordine N è un'operazione che consiste nel riordinare gli elementi di $\mathcal{X}$ in una matrice. L'unfolding di un tensore permette di riscrivere le operazioni tra tensori in equivalenti operazioni tra matrici, computazionalmente più vantaggiose. Inoltre, riarrangiando un *Dataset* in un tensore è possibile trovare relazioni/strutture nascoste riconoscendo caratteristiche (*pattern*) nei vari unfolding.

Esistono innumerevoli procedimenti per ottenere l'unfolding di un tensore, ciascuno ne preserva caratterestiche differenti. Di seguito viene trattato solo il *mode-n* unfolding, o unfolding modale, che verrà usato in seguito per determinare l'HOSVD. A loro volta esistono diversi unfolding modali, che però possono essere ricavati l'uno dall'altro mediante la moltiplicazione da destra per una matrice di permutazione. Sostanzialmente essi differiscono per l'ordine delle colonne, quindi l'utilizzo di un metodo piuttosto che di un altro in un algoritmo porta alla stessa soluzione.

Definizione 1.4 (***Mode-n* unfolding**). *Sia* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$. *Allora il suo mode-n unfolding è la matrice* $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$ *costruita in modo che l'elemento tensoriale* $(i_1, i_2, \ldots, i_N)$ *venga mappato nell'elemento matriciale* (i_n, j) *tale che*

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

In altre parole, il risultato del *mode-n* unfolding di un tensore $\mathcal{X}$ è una matrice, con le giuste dimensioni, avente le *n-esime* fibre di $\mathcal{X}$ sulle colonne. In alternativa, si può affermare che il *mode-n* unfolding di un tensore $\mathcal{X}$ è una matrice avente sulla *j-esima* riga gli elementi del subtensore di $\mathcal{X}$ ottenuto fissando l'*n-esima* coordinata uguale a j.

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

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

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

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

$$\boldsymbol{X}_{(3)} = \begin{bmatrix} 1 & 2 & \cdots & 11 & 12 \\ 13 & 14 & \cdots & 23 & 24 \end{bmatrix}.$$

È inoltre possibile riordinare gli elementi di un tensore in un vettore anziché in una matrice tramite l'operazione *vec*. Sia $\mathcal{X}$ come nell'esempio. Allora

$$vec(\mathcal{X}) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & \ldots 22 & 23 & 24 \end{bmatrix}^{\mathsf{T}}.$$

1.2 Il prodotto *Mode-n*

Il prodotto *mode-n*, o prodotto modale, è un'operazione che coinvolge un tensore, una matrice e un modo (n).

Definizione 1.5 (**Prodotto *Mode-n***). *Sia* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ *e* $\boldsymbol{U} \in \mathbb{R}^{J \times I_n}$ *allora il risultato del prodotto mode-n tra il tensore* $\mathcal{X}$ *e la matrice* $\boldsymbol{U}$ *viene indicato con* $(\mathcal{X} \times_n \boldsymbol{U})$ *ed è il tensore di dimensione* $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ *tale che :*

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

Ogni fibra *mode-n* di $\mathcal{X}$ viene moltiplicata per la matrice $\boldsymbol{U}$. Infatti il prodotto modale può essere espresso in forma matriciale come segue:

$$\mathcal{Y} = \mathcal{X} \times_n \boldsymbol{U} \text{ se e solo se } \boldsymbol{Y}_{(n)} = \boldsymbol{U}\boldsymbol{X}_{(n)}.$$

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

$$(\mathcal{X} \times_m \boldsymbol{A}) \times_n \boldsymbol{B} = (\mathcal{X} \times_n \boldsymbol{B}) \times_m \boldsymbol{A} \quad (m \neq n). \tag{1.1}$$

Mentre per lo stesso modo vale:

$$(\mathcal{X} \times_n \boldsymbol{A}) \times_n \boldsymbol{B} = \mathcal{X} \times_n (\boldsymbol{A}\boldsymbol{B}). \tag{1.2}$$

Il prodotto *mode-n* viene generalizzato nel prodotto multilineare.

Definizione 1.6 (**Prodotto Multilineare**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ un tensore e siano $\boldsymbol{M}_1 \in \mathbb{R}^{J_1 \times I_1}, \cdots, \boldsymbol{M}_N \in \mathbb{R}^{J_N \times I_N}$ N matrici; allora il risultato del prodotto multilineare tra $\mathcal{X}$ e le matrici $\boldsymbol{M}_k$ è il tensore $\mathcal{Y}$ di dimensione $J_1 \times \cdots \times J_N$ tale che :*

$$\mathcal{Y}(j_1, \cdots, j_N) = \sum_{i_1=1,\cdots,i_N=1}^{I_1,\cdots,I_N} x_{i_1,i_2,\cdots,i_N} \boldsymbol{M}_1(j_1, i_1)\boldsymbol{M}_2(j_2, i_2) \cdots \boldsymbol{M}_1(j_N, i_N).$$

Esso è equivalente a scrivere $\mathcal{Y} = \mathcal{X} \times_1 \boldsymbol{M}_1 \times_2 \boldsymbol{M}_2 \times_3 \cdots \times_N \boldsymbol{M}_N$ e come per il prodotto *mode-n* l'ordine è irrilevante.

Teorema 1.7. *Siano $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ e $\boldsymbol{M}_k \in R^{J_k \times I_k}$ per $k = 1, \ldots, N$. Sia $\mathcal{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ il tensore risultato del prodotto multilineare*

$$\mathcal{Y} = \mathcal{X} \times_1 \boldsymbol{M}_1 \times_2 \boldsymbol{M}_2 \times_2 \cdots \times_N \boldsymbol{M}_N,$$

allora

$$\boldsymbol{Y}_{(k)} = \boldsymbol{M}_k \cdot \boldsymbol{X}_{(k)} \cdot (\boldsymbol{M}_N \otimes \cdots \otimes \boldsymbol{M}_{k+1} \otimes \boldsymbol{M}_{k-1} \otimes \cdots \otimes \boldsymbol{M}_1)^{\mathsf{T}}.$$

Se $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_N$ sono quadrate non singolari, allora

$$\mathcal{X} = \mathcal{Y} \times_1 \boldsymbol{M}_1^{-1} \times_2 \boldsymbol{M}_2^{-1} \times_3 \cdots \times_N \boldsymbol{M}_N^{-1}.$$

1.3 Prodotto di Kronecker

Il prodotto di Kronecker è un'importante operazione matriciale che risulta utile per lo studio di procedimenti e algoritmi tensoriali.

Definizione 1.8 (**Prodotto di Kronecker**). *Siano* $\boldsymbol{A} \in \mathbb{R}^{I\times J}$ *e* $\boldsymbol{B} \in \mathbb{R}^{K\times L}$ *due matrici. Il risultato del loro prodotto di Kronecker viene indicato come* $\boldsymbol{A}\otimes\boldsymbol{B}$ *e corrisponde alla seguente matrice di dimensione* $(IK)\times(JL)$

$$\boldsymbol{A}\otimes\boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & a_{I2}\boldsymbol{B} & \cdots & a_{IJ}\boldsymbol{B} \end{bmatrix} \quad \boldsymbol{b} = \left[\boldsymbol{a}_1\otimes\boldsymbol{b}_1,\, \boldsymbol{a}_1\otimes\boldsymbol{b}_2,\, \boldsymbol{a}_1\otimes\boldsymbol{b}_3,\, \cdots,\, \boldsymbol{a}_J\otimes\boldsymbol{b}_{L-1},\, \boldsymbol{a}_J\otimes\boldsymbol{b}_L\right],$$

con $\boldsymbol{A} = [\boldsymbol{a}_1,\, \boldsymbol{a}_2,\, \cdots,\, \boldsymbol{a}_J]$ *e* $\boldsymbol{B} = [\boldsymbol{b}_1,\, \boldsymbol{b}_2,\, \cdots,\, \boldsymbol{b}_L]$.

1.4 Tensori di Rango uno

Definizione 1.9 (**Prodotto esterno**). *Siano* $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ *e* $\mathcal{Y} \in \mathbb{R}^{J_1\times\cdots\times J_M}$ *due tensori, il risultato del prodotto esterno tra* $\mathcal{X}$ *e* $\mathcal{Y}$ *viene indicato come* $\mathcal{X}\circ\mathcal{Y}$ *ed è un tensore* $\mathcal{C} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$*, di dimensione* $(N+M)$*, tale che*

$$\mathcal{C}(i_1,\cdots,i_N,j_1,\cdots,j_M) = \mathcal{X}(i_1,\cdots,i_N)\mathcal{Y}(j_1,\cdots,j_M).$$

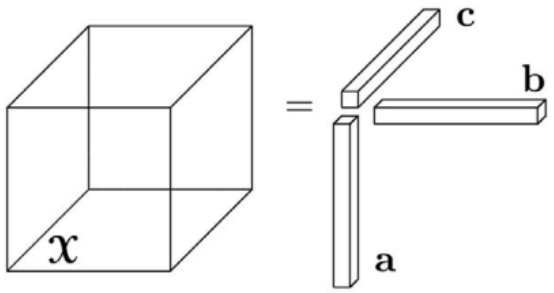

Figura 1.2: *Rappresentazione di un tensore di rango uno,* $\mathcal{X} = \boldsymbol{a}\circ\boldsymbol{b}\circ\boldsymbol{c}$.

Definizione 1.10 (**Tensore di rango uno**). *Un tensore* $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ *di ordine* N*, è detto di rango uno se può essere scritto come prodotto esterno di* N *vettori.*

$$\mathcal{X} = \boldsymbol{a}^{(1)}\circ\boldsymbol{a}^{(2)}\circ\cdots\circ\boldsymbol{a}^{(N)} \quad con \quad \boldsymbol{a}^{(k)} \in \mathbb{R}^{I_k} \quad per \quad k = 1,\cdots,N.$$

Equivalentemente:

$$x_{i_1,\cdots,i_N} = a^{(1)}(i_1)\cdot\cdots\cdot a^{(N)}(i_N) \quad per\ 1 \le i_k \le I_k \quad e\ k = 1,\cdots,N.$$

Esiste un importante connessione tra i tensori di rango uno e il prodotto di Kronecker. Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore di rango uno come nella Definizione 1.10; allora per ogni $k = 1, \cdots, N$ l'unfolding modale $\boldsymbol{X}_{(k)}$ è una matrice di rango uno che soddisfa la seguente relazione :

$$\boldsymbol{X}_{(k)} = \boldsymbol{a}^{(k)} \cdot (\boldsymbol{a}^{(N)} \otimes \cdots \otimes \boldsymbol{a}^{(k+1)} \otimes \boldsymbol{a}^{(k-1)} \otimes \cdots \otimes \boldsymbol{a}^{(1)})^{\mathsf{T}}. \tag{1.3}$$

È importante osservare che qualsiasi tensore può essere scritto come somma di tensori di rango uno. Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un qualsiasi tensore di dimensione N; definiamo $\boldsymbol{e}_k(i)$, per $i = 1, \cdots, I_k$ e $k = 1, \cdots, N$, come il vettore della base canonica di $\mathbb{R}^{I_k}$ avente un uno in posizione i, allora vale

$$\mathcal{X} = \sum_{i_1=1,\cdots,i_N=1}^{I_1,\cdots,I_N} x_{i_1,\cdots,i_N}(\boldsymbol{e}_1(i_1) \circ \cdots \circ \boldsymbol{e}_N(i_N)). \tag{1.4}$$

Capitolo 2

Decomposizione CP e Rango di un Tensore

Nel 1927, Hitchcock propose l'idea di esprimere un tensore come somma di un numero finito di tensori rango uno, si veda, per esempio [1]. Quest'idea venne ripresa negli anni successivi e divenne popolare nel 1970 sotto il nome di CANDECOMP (decomposizione canonica) nella versione proposta da Carroll e Chang, e PARAFAC (fattori paralleli) nella versione proposta da Harshman. Nel 2000, Kiers si riferì alla decomposizione CANDECOMP/PARAFAC come CP. Per la descrizione di tali decomposizioni in questo capitolo vengono trattati i tensori di terzo ordine in campo reale ma i risultati ottenuti possono essere generalizzarti all'ordine N e al campo complesso.

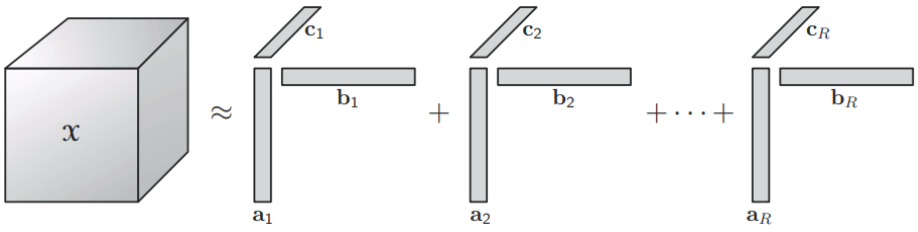

Figura 2.1: *Rappresentazione della approssimazione mediante decomposizione CP di* $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

Formalmente, la decomposizione CP fattorizza un tensore come somma di tensori di rango uno. Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ un tensore di ordine 3. Vorremmo approssimarlo mediante:

$$\mathcal{X} \approx \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r, \tag{2.1}$$

dove R è un intero positivo e $\boldsymbol{a}_r \in \mathbb{R}^I$, $\boldsymbol{b}_r \in \mathbb{R}^J$ e $\boldsymbol{c}_r \in R^K$ per $r = 1, \cdots, R$. Equivalentemente,

$$x_{i,j,k} \approx \sum_{r=1}^{R} a_r(i) b_r(j) c_r(k) \quad \text{per } i = 1, \cdots, I;\ j = 1, \cdots, J;\ k = 1, \cdots, K.$$

È immediato costruire le matrici della decomposizione, che contengono i vettori della scomposizione sulle colonne; $\boldsymbol{A} = [\,\boldsymbol{a}_1, \cdots, \boldsymbol{a}_R\,] \in \mathbb{R}^{I \times R}$, $\boldsymbol{B} = [\,\boldsymbol{b}_1, \cdots, \boldsymbol{b}_R\,] \in \mathbb{R}^{J \times R}$ e $\boldsymbol{C} = [\,\boldsymbol{c}_1, \cdots, \boldsymbol{c}_R\,] \in \mathbb{R}^{K \times R}$. Un'altra notazione molto utilizzata per indicare la decomposizione CP è la seguente:

$$\mathcal{X} \approx [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r. \tag{2.2}$$

In molte applicazioni, anche nella stessa implementazione della decomposizione, è necessario richiedere che $\boldsymbol{A}$, $\boldsymbol{B}$ e $\boldsymbol{C}$ abbiano colonne normalizzate, di conseguenza si introduce un "peso", $\boldsymbol{\lambda} \in \mathbb{R}^R$, tale che:

$$\mathcal{X} \approx [\![\boldsymbol{\lambda}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r, \tag{2.3}$$

con $\|\boldsymbol{a}_r\| = \|\boldsymbol{b}_r\| = \|\boldsymbol{c}_r\| = 1$ per $r = 1, \ldots, R$.

Nel generico caso di un tensore di ordine N, vale:

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

dove R è un intero positivo e $\boldsymbol{A}^{(k)} \in \mathbb{R}^{I_k \times R}$ per $k = 1, \cdots, N$.

2.1 Rango di un Tensore

Definizione 2.1 (**Rango di un Tensore**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore, il rango di $\mathcal{X}$, indicato come $rank(\mathcal{X})$, è il minimo numero intero positivo di tensori di rango uno la cui somma è esattamente $\mathcal{X}$.*

$$rank(\mathcal{X}) = \arg\min_{R \in \mathbb{N}} \left\{ \exists\, \boldsymbol{A}^{(k)} \in \mathbb{R}^{I_k \times R} \; per \;\; k = 1, \cdots, N \,|\, \mathcal{X} \equiv [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!] \right\}.$$

Si osserva che il termine "esattamente" presente nella definizione ha un significato matematico ben preciso; infatti esso si riferisce a una decomposizione CP esatta, che differisce dall'equazione (2.2) per la presenza dell'uguaglianza (=) al posto del simbolo di approssimazione (≈). Dato un tensore, la sua decomposizione CP esatta avente il

minimo numero di tensori di rango uno è detta decomposizione di rango. Inoltre il rango di un tensore è invariante rispetto a permutazioni dei modi del tensore.

Non esiste un algoritmo efficiente in grado di ricavare la decomposizione di rango di un tensore dato. È possibile dimostrare che il calcolo del rango di un tensore è un problema NP-arduo, esso infatti ha numerose complicazioni, come la dipendenza dal campo in cui si considera il tensore.

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

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

Questo è un tensore di rango tre in $\mathbb{R}$, mentre è due in $\mathbb{C}$. Infatti se $\mathcal{X} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$; allora in $\mathbb{R}$

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

mentre su $\mathbb{C}$

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

La definizione di rango di un tensore porta con sé altre due nozioni fondamentali di carattere più generale: il rango massimo e il rango tipico.

Definizione 2.2 (**Rango massimo e rango tipico**). *Fissati $N \in \mathbb{N}$ e $I_1, \cdots, I_N \in \mathbb{N}$; allora si definisce come:*
(1) Rango massimo, il massimo rango ottenibile tale che esiste un tensore di dimensione $I_1 \times \cdots \times I_N$ avente quel rango.
(2) Rango tipico, un possibile rango che un tensore di dimensione $I_1 \times \cdots \times I_N$ può assumere con probabilità maggiore di zero.

Ad esempio, per le matrici $I \times J$ il rango massimo e tipico coincidono e sono uguali al $\min\{I, J\}$. Tuttavia per i tensori di ordine maggiore possono essere diversi. Infatti, il rango massimo di una classe di tensori è unico, mentre possono esistere più ranghi tipici (in $\mathbb{R}$). Ad esempio, considerando i tensori reali $2 \times 2 \times 2$ e supponendo che i loro elementi siano distribuiti secondo una distribuzione normale standard é possibile verificare sperimentalmente che il 79% ha rango due e il 21% ha rango tre; esistono anche tensori $2 \times 2 \times 2$ di rango uno ma la loro probabilità è nulla.

Fissati $I_1, \cdots, I_N$ non esiste nessuna formula generica per determinare il rango massimo dei tensori di questa dimensione, esso è noto soltanto per alcuni casi specifici, perciò

si cerca un limite dall'alto. Per il rango massimo dei tensori di ordine tre vale la seguente stima dall'alto. Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, allora:

$$rank(\mathcal{X}) \leq \min\{IJ, IK, JK\}. \tag{2.4}$$

2.2 Unicità

Una proprietà dei tensori di ordine alto è che la loro decomposizione di rango è spesso unica. Nei tensori di ordine basso, come le matrici, questo non accade. Sia $\boldsymbol{X} \in \mathbb{R}^{I\times J}$ tale che $\boldsymbol{X} = \boldsymbol{AB}$ allora, presa una qualsiasi matrice $\boldsymbol{W}$ non singolare delle giuste dimensioni, vale $\boldsymbol{X} = (\boldsymbol{AW})(\boldsymbol{W}^{-1}\boldsymbol{B})$. L'unicità della decomposizione viene esposta prendendo come riferimento il testo *Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics* di Kruskal [2].

Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ un tensore di ordine tre di rango R tale che

$$\mathcal{X} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r.$$

La decomposizione CP di $\mathcal{X}$ è unica se quella appena proposta è la sola combinazione possibile di tensori di rango uno la cui somma fa esattamente $\mathcal{X}$, a meno di permutazioni o riscalamento. L'invarianza per permutazioni deriva dal fatto che la somma è commutativa e quindi si possono riordinare le componenti di rango uno;

$$\mathcal{X} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = [\![\boldsymbol{A\Pi}, \boldsymbol{B\Pi}, \boldsymbol{C\Pi}]\!] \text{ con } \boldsymbol{\Pi} \text{ matrice di permutazione } R\times R \text{ qualsiasi.}$$

Mentre l'invarianza per riscalamento deriva dal fatto che ciascun vettore può essere moltiplicato indipendentemente per uno scalare;

$$\mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r = \sum_{r=1}^{R} (\alpha_r \boldsymbol{a}_r) \circ (\beta_r \boldsymbol{b}_r) \circ (\gamma_r \boldsymbol{c}_r) \text{ con } \alpha_r\beta_r\gamma_r = 1 \text{ per } r = 1, \cdots, R.$$

Riassumendo, $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ tensore di ordine tre di rango R ha un'unica decomposizione CP, ($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$), se tutte le sue decomposizioni possono essere scritte come

$$\mathcal{X} = [\![\boldsymbol{A\Pi\Lambda}, \boldsymbol{B\Pi L}, \boldsymbol{C\Pi M}]\!], \tag{2.5}$$

con $\boldsymbol{\Pi} \in \mathbb{R}^{R\times R}$ matrice di permutazione e $\boldsymbol{\Lambda, L, M} \in \mathbb{R}^{R\times R}$ matrici diagonali tali che il prodotto $\boldsymbol{\Lambda LM}$ è la matrice identità. Due decomposizioni di rango di $\mathcal{X}$ che differiscono per il prodotto di una matrice di permutazione o di tre matrici di riscalamento si dicono equivalenti. Con la notazione e le definizioni introdotte, $\mathcal{X}$ ha un'unica decomposizione CP se e solo se tutte le sue decomposizioni di rango sono equivalenti.

Per i tensori di rango uno vale il seguente risultato.

Teorema 2.3. *Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ un tensore di rango uno, $R = 1$, tale che $\mathcal{X} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = [\![\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}]\!]$ e siano $I_0 = rank(\boldsymbol{A})$, $J_0 = rank(\boldsymbol{B})$, $K_0 = rank(\boldsymbol{C})$.*
Se $I_0 + J_0 + K_0 \geq 3$, allora $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ e $\left(\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}\right)$ sono equivalenti.

Dimostrazione. Siccome $R = 1$, le sei matrici $\boldsymbol{A}, \boldsymbol{B}, \ldots$ sono vettori colonna, e l'unica matrice di permutazione possibile è la matrice identità 1×1. Dall'ipotesi sulla somma dei ranghi otteniamo che $I_0 = J_0 = K_0 = 1$, quindi tutti e sei i vettori sono diversi da quello nullo. È quindi elementare mostrare che $\bar{\boldsymbol{A}}$ è proporzionale a $\boldsymbol{A}$, analogo per $\boldsymbol{B}$ e $\boldsymbol{C}$. □

Per ottenere un risultato analogo al precedente per i tensori di rango maggiore di uno bisogna prima introdurre il concetto di *k-rango* di una matrice.

Definizione 2.4 (**K-Rango**). *Sia $\boldsymbol{A} \in \mathbb{R}^{I\times J}$ una matrice, si dice k-rango di $\boldsymbol{A}$, e si denota k_A, il massimo numero k tale che qualsiasi insieme di k colonne di $\boldsymbol{A}$ sono linearmente indipendenti.*

Per capire immediatamente la differenza con la classica definizione di rango di matrice è sufficiente il seguente esempio.

Esempio: Sia $\boldsymbol{A} \in \mathbb{R}^{2\times 3}$ tale che:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Vale che $rank(\boldsymbol{A}) = 2$, ma $k_A = 1$ perché le prime due colonne sono linearmente dipendenti.

Teorema 2.5. *Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ un tensore di rango R, tale che $\mathcal{X} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = [\![\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}]\!]$. Se $k_A + k_B + k_C \geq 2R + 2$, allora $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ e $\left(\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}\right)$ sono equivalenti.*

Il Teorema 2.5 proposto da Kruskal venne generalizzato ai tensori di ordine N da Sidiropoulos e Bro in [4].

Sia $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ un tensore di ordine N e rango R, la cui decomposizione CP è

$$\mathcal{X} = [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!].$$

Condizione sufficiente per l'unicità è

$$\sum_{n=1}^{N} k_{A^{(n)}} \geq 2R + (N - 1). \tag{2.6}$$

Tutti i risultati precedenti sono condizioni sufficienti per l'unicità della decomposizione CP, ma per i tensori di rango $R = 2$ o $R = 3$ l'equazione (2.6) è anche condizione necessaria.

2.3 Approssimazione di Rango Basso

Un problema strettamente legato alla decomposizione CP è quello di trovare la migliore approssimazione di rango k di un qualsiasi tensore. Dato $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ tensore di ordine tre, vogliamo risolvere:

$$\mathcal{Y} = \underset{\substack{\mathcal{Y}\in\mathbb{R}^{I\times J\times K}\\ rank(\mathcal{Y})=k}}{\arg\min} \{\|\mathcal{X}-\mathcal{Y}\|\}. \tag{2.7}$$

Per le matrici, la migliore approssimazione di rango k è data dai primi k fattori dominanti della SVD. Ad esempio, sia $\boldsymbol{A}$ una matrice di rango R con la seguente scomposizione SVD:

$$\boldsymbol{A} = \sum_{r=1}^{R} \sigma_r \boldsymbol{u}_r \circ \boldsymbol{v}_r, \quad \text{con} \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R > 0.$$

Allora, l'approssimazione di rango k che minimizza $\|\boldsymbol{A}-\boldsymbol{B}\|$ è

$$\boldsymbol{B} = \sum_{r=1}^{k} \sigma_r \boldsymbol{u}_r \circ \boldsymbol{v}_r.$$

Purtroppo per i tensori di ordine superiore a due non esiste un risultato simile.

Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ un tensore di ordine tre di rango R con la seguente decomposizione CP:

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r.$$

La somma di k di questi tensori di rango uno non fornisce la migliore approssimazione di $\mathcal{X}$ di rango k. Inoltre se

$$\mathcal{X}_{k+1} = \sum_{r=1}^{k+1} \lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r,$$

è la migliore approssimazione di $\mathcal{X}$ di rango $k+1$, non vuol dire che

$$\mathcal{X}_k = \sum_{r=1}^{k} \lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r,$$

sia la migliore approssimazione di $\mathcal{X}$ di rango k. Questo fatto suggerisce che le componenti della migliore approssimazione di X di rango k vanno cercate simultaneamente e non sequenzialmente.

Inoltre la migliore approssimazione di rango k potrebbe non esistere.

Definizione 2.6 (**Tensore Degenere**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ un tensore di rango R. $\mathcal{X}$ si dice degenere se può essere approssimato in maniera arbitrariamente precisa da un tensore di rango k con $k < R$.*

Esempio Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ un tensore di rango tre definito come

$$\mathcal{X} = \boldsymbol{a}_1 \circ \boldsymbol{b}_1 \circ \boldsymbol{c}_2 + \boldsymbol{a}_1 \circ \boldsymbol{b}_2 \circ \boldsymbol{c}_1 + \boldsymbol{a}_2 \circ \boldsymbol{b}_1 \circ \boldsymbol{c}_1,$$

dove $\boldsymbol{A} \in \mathbb{R}^{I\times 2}$, $\boldsymbol{B} \in \mathbb{R}^{J\times 2}$ e $\boldsymbol{C} \in \mathbb{R}^{K\times 2}$, e ciascuna ha colonne linearmente indipendenti. Questo tensore può essere approssimato da un tensore di rango due della forma:

$$\mathcal{Y} = \alpha \left(\boldsymbol{a}_1 + \frac{1}{\alpha}\boldsymbol{a}_2\right) \circ \left(\boldsymbol{b}_1 + \frac{1}{\alpha}\boldsymbol{b}_2\right) \circ \left(\boldsymbol{c}_1 + \frac{1}{\alpha}\boldsymbol{c}_2\right) - \alpha\, \boldsymbol{a}_1 \circ \boldsymbol{b}_1 \circ \boldsymbol{c}_1.$$

In particolare,

$$\|\mathcal{X} - \mathcal{Y}\| = \frac{1}{\alpha} \left\| \boldsymbol{a}_2 \circ \boldsymbol{b}_2 \circ \boldsymbol{c}_1 + \boldsymbol{a}_2 \circ \boldsymbol{b}_1 \circ \boldsymbol{c}_2 + \boldsymbol{a}_1 \circ \boldsymbol{b}_2 \circ \boldsymbol{c}_2 + \frac{1}{\alpha}\boldsymbol{a}_2 \circ \boldsymbol{b}_2 \circ \boldsymbol{c}_2 \right\|,$$

che può essere resa piccola a piacere. Quindi la migliore approssimazione di rango due di $\mathcal{X}$ non esiste.

2.4 Aspetti Computazionali

Come detto precedentemente non esiste un algoritmo finito in grado di determinare il rango di un tensore. Esistono però algoritmi che, fissato un numero R, calcolano la decomposizione CP di rango R di un tensore, se esiste. L'algoritmo più utilizzato, del quale verrà esposta solo l'idea senza entrare nei dettagli, è il metodo iterativo ALS (Alternating Least Squares).

Sia $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ un tensore di ordine tre e sia R fissato, lo scopo è trovare $\mathcal{Y}$ che risolva

$$\min_{\mathcal{Y}\in\mathbb{R}^{I\times J\times K}} \|\mathcal{X} - \mathcal{Y}\| \quad \text{con} \quad \mathcal{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r = [\![\boldsymbol{\lambda}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]. \tag{2.8}$$

Il metodo ALS fissa $\boldsymbol{B}$ e $\boldsymbol{C}$ e risolve l'equazione (2.8) per $\boldsymbol{A}$, quindi fissa $\boldsymbol{A}$ (trovata precedentemente) e $\boldsymbol{C}$ e risolve per $\boldsymbol{B}$, quindi fissa $\boldsymbol{A}$ e $\boldsymbol{B}$ e risolve per $\boldsymbol{C}$ e continua a ripetere l'intera procedura fino a quando qualche criterio di convergenza è soddisfatto; in generale può essere implementato per i tensori di ordine N.

Alcune possibili condizioni di arresto sono: il residuo $\|\mathcal{X} - \mathcal{Y}\|$ diventa più piccolo di una certa tolleranza, le matrici della scomposizione rimangono invariate, oppure viene superato un numero massimo predefinito di iterazioni. Il metodo ALS è semplice da implementare, ma può richiedere molte iterazioni per convergere.

Nel calcolo del rango di un tensore è quindi importante la scelta del numero R di componenti di rango uno. Per ovviare a tale problema viene eseguita la decomposizione CP facendo aumentare progressivamente il numero di componenti di rango uno, $R = 1, 2, \cdots$, fino a quando non si trova una decomposizione soddisfacente.

Capitolo 3

Decomposizione HOSVD

L'HOSVD, High Order Singular Value Decomposition, detta anche decomposizione di Tucker, poiché introdotta da Tucker nel 1963, è una decomposizione tensoriale considerata una generalizzazione a più dimensioni della SVD matriciale, [1]. L'obiettivo di questa decomposizione è scrivere un tensore $\mathcal{X}$ di dimensione N come il prodotto multilineare di un tensore $\mathcal{G}$, detto *core tensor*, ed N diverse matrici $\boldsymbol{A}^{(n)}$, una per ogni modo.

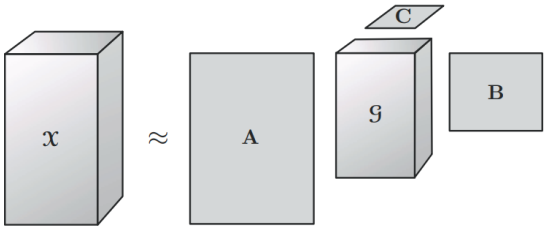

Figura 3.1: *Rappresentazione della HOSVD di* $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. Si vuole ottenere una scrittura del tipo:

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{p,q,r} \boldsymbol{a}_p \circ \boldsymbol{b}_q \circ \boldsymbol{c}_r, \tag{3.1}$$

dove $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$, $\boldsymbol{A} \in \mathbb{R}^{I \times P}$, $\boldsymbol{B} \in \mathbb{R}^{J \times Q}$ e $\boldsymbol{C} \in \mathbb{R}^{K \times R}$. Equivalentemente:

$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{p,q,r}\, a_{i,p} b_{j,q} c_{k,r}, \quad \text{per } i = 1, \ldots, I;\ j = 1, \ldots, J;\ k = 1, \ldots, K.$$

Solitamente le matrici $\boldsymbol{A}$, $\boldsymbol{B}$ e $\boldsymbol{C}$ sono prese ortogonali e possono essere interpretate come le componenti principali lungo i tre diversi modi.

L'HOSVD viene indicata come segue:

$$\mathcal{X} \approx [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \tag{3.2}$$

La scelta di questa notazione, praticamente uguale a quella utilizzata nel Capitolo 2 per la decomposizione CP, è voluta. Infatti la decomposizione CP è un particolare caso della HOSVD, dove $P = Q = R$ e il core tensor $\mathcal{G}$ è diagonale.

La decomposizione (3.1) può essere espressa in forma matriciale, utilizzando il Teorema 1.7, come segue:

$$\begin{aligned} \boldsymbol{X}_{(1)} &\approx \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^{\mathsf{T}}, \\ \boldsymbol{X}_{(2)} &\approx \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^{\mathsf{T}}, \\ \boldsymbol{X}_{(3)} &\approx \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})^{\mathsf{T}}. \end{aligned} \tag{3.3}$$

Il modello di Tucker può essere generalizzato a tensori N-dimensionali come

$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \cdots \times_N \boldsymbol{A}^{(N)} = \left[\!\left[\mathcal{G}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}\right]\!\right]. \tag{3.4}$$

Equivalentemente:

$$x_{i_1,i_2,\cdots,i_N} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1,r_2\cdots,r_N} a^{(1)}_{i_1,r_1} a^{(2)}_{i_2,r_2} \cdots a^{(N)}_{i_N,r_N} \quad \text{per } i_n = 1, \cdots, I_n; \quad n = 1, \cdots, N.$$

La forma matricale della (3.4) è

$$\boldsymbol{X}_{(n)} = \boldsymbol{A}^{(n)}\boldsymbol{G}_{(n)} \left(\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{(n+1)} \otimes \boldsymbol{A}^{(n-1)} \otimes \cdots \otimes \boldsymbol{A}^{(1)}\right)^{\mathsf{T}}.$$

Seguendo la Sezione 12.4 in [3], spieghiamo perché la decomposizione Tucker è considerata la generalizzazione della SVD.

Sia $\boldsymbol{X} \in \mathbb{R}^{I \times J}$. La sua SVD,

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} = \sum_{r=1}^{R} \sigma_r \boldsymbol{u}_r \circ \boldsymbol{v}_r,$$

può anche essere scritta come $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{X} = \boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}$. Da tale scrittura è facile osservare che le righe di $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{X}$ sono tra loro ortogonali e monotone decrescenti nella norma, infatti:

$$\boldsymbol{U}^{\mathsf{T}}\boldsymbol{X} = \begin{bmatrix} \sigma_1 \boldsymbol{v}_1^{\mathsf{T}} \\ \vdots \\ \sigma_n \boldsymbol{v}_n^{\mathsf{T}} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3.5}$$

Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ un tensore di ordine tre e consideriamo l'SVD di ogni unfolding modale:

$$\boldsymbol{U}_1^{\mathsf{T}}\boldsymbol{X}_{(1)} = \boldsymbol{\Sigma}_1\boldsymbol{V}_1^{\mathsf{T}}, \quad \boldsymbol{U}_2^{\mathsf{T}}\boldsymbol{X}_{(2)} = \boldsymbol{\Sigma}_2\boldsymbol{V}_2^{\mathsf{T}}, \quad \boldsymbol{U}_3^{\mathsf{T}}\boldsymbol{X}_{(3)} = \boldsymbol{\Sigma}_3\boldsymbol{V}_3^{\mathsf{T}}. \tag{3.6}$$

Da queste tre scomposizioni si possono definire tre prodotti modali:

$$\mathcal{G}^{(1)} = \mathcal{X} \times_1 \boldsymbol{U}_1^\mathsf{T}, \quad \mathcal{G}^{(2)} = \mathcal{X} \times_2 \boldsymbol{U}_2^\mathsf{T}, \quad \mathcal{G}^{(3)} = \mathcal{X} \times_3 \boldsymbol{U}_3^\mathsf{T}. \tag{3.7}$$

Utilizzando il Teorema 1.7, abbiamo i seguenti unfolding:

$$\mathcal{G}_{(1)}^{(1)} = \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\mathsf{T}, \quad \mathcal{G}_{(2)}^{(2)} = \boldsymbol{\Sigma}_2 \boldsymbol{V}_2^\mathsf{T}, \quad \mathcal{G}_{(3)}^{(3)} = \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\mathsf{T}.$$

Richiamando l'equazione (3.5) e ricordando che le righe degli unfolding contengono gli elementi dei sottotensori, è facile dimostrare che

$$\|\mathcal{G}^{(1)}(i,:,:)\|_F = \sigma_i(\boldsymbol{X}_{(1)}), \quad i = 1:I,$$

$$\|\mathcal{G}^{(2)}(:,j,:)\|_F = \sigma_i(\boldsymbol{X}_{(2)}), \quad j = 1:J,$$

$$\|\mathcal{G}^{(3)}(:,:,k)\|_F = \sigma_i(\boldsymbol{X}_{(3)}), \quad k = 1:K.$$

Unendo i tre prodotti dell'equazione (3.7) in un solo prodotto multilineare otteniamo

$$\mathcal{G} = \mathcal{X} \times_1 \boldsymbol{U}_1^\mathsf{T} \times_2 \boldsymbol{U}_2^\mathsf{T} \times_3 \boldsymbol{U}_3^\mathsf{T}.$$

Poiché le matrici $\boldsymbol{U}_i$ sono ortogonali, possiamo utilizzare l'ultima implicazione del Teorema 1.7 e scrivere

$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3.$$

Teorema 3.1 (**HOSVD**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore e siano*

$$\boldsymbol{X}_{(k)} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^\mathsf{T} \quad \textit{per } k = 1, \ldots, N\,, \tag{3.8}$$

le SVD di tutti i suoi unfolding modali con $\boldsymbol{U}_k \in \mathbb{R}^{I_k \times I_k}$, allora una HOSVD è data da

$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \cdots \times_N \boldsymbol{U}_N,$$

dove $\mathcal{G} = \mathcal{X} \times_1 \boldsymbol{U}_1^\mathsf{T} \times_2 \boldsymbol{U}_2^\mathsf{T} \times_3 \cdots \times_N \boldsymbol{U}_N^\mathsf{T}$. La formulazione (3.1) *è equivalente a*

$$\mathcal{X} = \sum_{j_1=1,\cdots,j_N=1}^{I_1,\cdots,I_N} \mathcal{G}(j_1, \ldots, j_N) \boldsymbol{U}_1(:, j_1) \circ \cdots \circ \boldsymbol{U}_N(:, j_N),$$

$$\mathcal{X}(i_1, \ldots, i_N) = \sum_{j_1=1,\cdots,j_N=1}^{I_1,\cdots,I_N} \mathcal{G}(j_1, \ldots, j_N) \boldsymbol{U}_1(i_1, j_1) \cdots \boldsymbol{U}_N(i_N, j_N),$$

$$vec(\mathcal{X}) = (\boldsymbol{U}_d \otimes \cdots \otimes \boldsymbol{U}_1) vec(\mathcal{G}).$$

Inoltre valgono,

$$\langle \mathcal{G}_{i_n=\alpha}, \mathcal{G}_{i_n=\beta} \rangle = 0 \quad \textit{quando} \quad \alpha \neq \beta \quad \textit{per} \quad n = 1, \ldots, N, \tag{3.9}$$

dove, $\mathcal{G}_{i_n=\alpha}$ è il subtensore di $\mathcal{G}$ ottenuto fissanto l'n-esima coordinata uguale ad α; e

$$\|\boldsymbol{G}_{(k)}(i,:)\|_F = \sigma_i(\boldsymbol{X}_{(k)}), \quad \textit{per } i = 1, \cdots rank(\boldsymbol{X}_{(k)}) \quad e \ k = 1, \cdots, N. \tag{3.10}$$

Dimostrazione. Verifichiamo soltanto l'uguaglianza (3.10). Si noti che

$$\boldsymbol{G}_{(k)} = \boldsymbol{U}_k^{\mathsf{T}} \boldsymbol{X}_{(k)} (\boldsymbol{U}_N^{\mathsf{T}} \otimes \cdots \otimes \boldsymbol{U}_{k+1}^{\mathsf{T}} \otimes \boldsymbol{U}_{k-1}^{\mathsf{T}} \otimes \cdots \otimes \boldsymbol{U}_1^{\mathsf{T}})^{\mathsf{T}} = \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T (\boldsymbol{U}_d \otimes \cdots \otimes \boldsymbol{U}_{k-1} \otimes \cdots \otimes \boldsymbol{U}_1).$$

Ne segue che le righe di $\boldsymbol{G}_{(k)}$ sono tra loro ortogonali e che i valori singolari di $\boldsymbol{X}_{(k)}$ sono le norme euclidee di queste righe. □

3.1 Mancanza di Unicità

La decomposizione HOSVD non è unica. Sia $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ un tensore scritto come nell'equazione (3.1) e siano $\boldsymbol{P} \in \mathbb{R}^{P \times P}$, $\boldsymbol{Q} \in \mathbb{R}^{Q \times Q}$ e $\boldsymbol{W} \in \mathbb{R}^{R \times R}$ matrici ortogonali. Allora vale

$$[\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = [\![\mathcal{G} \times_1 \boldsymbol{P} \times_2 \boldsymbol{Q} \times_3 \boldsymbol{W}; \boldsymbol{A}\boldsymbol{P}^{\mathsf{T}}, \boldsymbol{B}\boldsymbol{Q}^{\mathsf{T}}, \boldsymbol{C}\boldsymbol{W}^{\mathsf{T}}]\!] .$$

In altre parole possiamo modificare il core tensor moltiplicandolo per matrici invertibili, senza cambiare la struttura del tensore.

Questa libertà ci permette di scegliere scomposizioni con un core tensor la cui struttura è il più semplice ed efficace possibile. Una tecnica molto utilizzata è quella di rendere il core tensor sparso, ovvero con il maggior numero di zeri possibile. Lavorare con un tensore sparso è computazionalmente molto efficiente in quanto c'è un risparmio di memoria, ci sono meno dati non zero da salvare, e i calcoli sono più veloci. Soprattutto in ambiti dove vengono utilizzati tensori con dimensioni molto grandi, come nel Data Mining, è fondamentale utilizzare queste tecniche per diminuire il costo computazionale degli algoritmi.

3.2 n-Rango

Come riportato nella Sezione 3.1, lavorare con tensore dalle grandi dimensioni è computazionalmente molto dispendioso. Approssimare un tensore con uno dalle dimensioni più modeste può portare ad un significativo risparmio di memoria. Per formalizzare questo concetto viene introdotta la nozione di n-rango.

Definizione 3.2 (**n-Rango**). *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ un tensore di ordine N. Allora l'n-rango di $\mathcal{X}$, denotato come $rank_n(\mathcal{X})$, è il rango colonna di $\boldsymbol{X}_{(n)}$:*

$$rank_n(\mathcal{X}) = rank(\boldsymbol{X}_{(n)})$$

In altre parole, l'n-rango di un tensore $\mathcal{X}$ è la dimensione dello spazio vettoriale generato dalle fibre *mode-n* e non deve essere confuso con la nozione di rango di un

tensore, vedi Sezione 2.1. Se denotiamo $R_n = rank_n(\mathcal{X})$ per $n = 1, \cdots, N$, allora possiamo dire che $\mathcal{X}$ è un tensore di rango *mode-n* $(R_1, R_2, \cdots, R_N)$. Vale $R_n \leq I_n$ per tutti $n = 1, \cdots, N$.

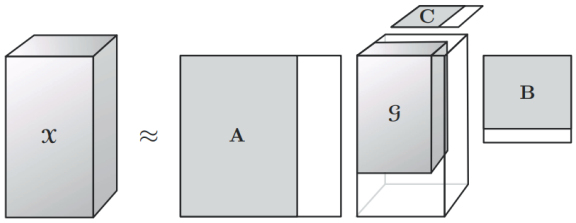

Figura 3.2: *Rappresentazione della HOSVD troncata di* $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

Dato un tensore $\mathcal{X}$, applicando il Teorema 3.1, si ottiene una decomposizione esatta di Tucker di rango $(R_1, R_2, \ldots, R_N)$, dove $R_n = rank_n(\mathcal{X})$. Tuttavia, è possibie calcolare una decomposizione HOSVD di rango $(r_1, r_2, \ldots, r_N)$, dove $r_n < rank_n(\mathcal{X})$ per uno o più n; essa è un'approssimazione, ed è chiamata HOSVD troncata. In molti algoritmi, lavorare con i fattori dell'HOSVD troncata, piuttosto che con il tensore completo $\mathcal{X}$, diminuisce il costo computazionale del metodo, infatti utilizzare un core tensor dalle piccole dimensioni (rispetto a quelle di $\mathcal{X}$) porta a un risparmio della memoria.

3.3 Algoritmo e Aspetti Computazionali

Nel 1966 Tucker propose diversi algoritmi per il calcolo della HOSVD nel caso dei tensori di ordine tre, e successivamente vennero generalizzati al caso N dimensionale, [1]. Il Teorema 3.1 suggerisce una prima implementazione della decomposizione.

Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore di rango *mode-n* $(R_1, R_2, \ldots, R_N)$, e siano $r_1, \ldots, r_N$ dei numeri naturali tali che $0 < r_k \leq R_k$ per $k = 1, \ldots, N$:

Algorithm 1 HOSVD esatta e troncata

function HOSVD$(\mathcal{X}, r_1, \ldots, r_N)$

  for $n = 1, \ldots, N$ **do**

    $\boldsymbol{U}_n \leftarrow$ i primi r_n vettori singolari sinistri della SVD di $\boldsymbol{X}_{(n)}$

  end for

  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{U}_1^{\mathsf{T}} \times_2 \cdots \times_N \boldsymbol{U}_N^{\mathsf{T}}$

return $\mathcal{G}, \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N$

Nel caso in cui $r_k = R_K$ per $k = 1, \ldots, N$ l'Algoritmo 1 è un esempio di un possibile modo di calcolare una HOSVD esatta. Se $r_k < R_k$ per uno o più k, l'Algoritmo 1 calcola una HOSVD troncata, e richiede soltanto il calcolo delle prime r_k triplette della SVD dei vari unfolding $\boldsymbol{X}_{(k)}$ e non di tutte le SVD complete, che è un procedimento computazionalmente molto dispendioso.

Oltre al tema del costo computazionale, nell'utilizzo dell'HOSVD troncata è fondamentale la precisione dell'approssimazione; L. De Lathauwer, B. De Moor e J. Vandewalle in [5] riportano il seguente risultato.

Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore di rango *mode-n* $(R_1, R_2, \ldots, R_N)$; definiamo come $\overline{\mathcal{X}}$ il tensore risultante dalla HOSVD troncata, ottenuto scartando i vettori singolari sinistri corrispondenti ai valori singolari più piccoli della SVD di $\boldsymbol{X}_{(k)}$, $\sigma^{(k)}_{r_k+1}, \ldots, \sigma^{(k)}_{R_k}$ per dati valori fissati r_k, con $1 \leq r_k < R_k$ e $k = 1, \ldots, N$. Allora vale

$$\left\|\mathcal{X} - \overline{\mathcal{X}}\right\|^2 \leq \sum_{i_1=r_1+1}^{R_1} \left(\sigma^{(1)}_{i_1}\right)^2 + \sum_{i_2=r_2+1}^{R_2} \left(\sigma^{(2)}_{i_2}\right)^2 + \cdots + \sum_{i_N=r_N+1}^{R_N} \left(\sigma^{(N)}_{i_N}\right)^2.$$

Questa stima dall'alto dà un'idea di come l'errorre dell'approssimazione di $\mathcal{X}$ fornita dalla HOSVD, calcolata come nell'Algoritmo 1, dipende fortemente dal valore del troncamento e da quello degli autovalori dei vari unfolding. Sebbene questo algoritmo fornisca un'opzione per calcolare la Tucker troncata di un tensore, quella ottenuta non è la migliore approssimazione di $\mathcal{X}$ con uno specifico rango *mode-n*.

3.4 La migliore approssimazione di rango $(r_1, \ldots, r_N)$

In questa sezione viene approfondito il tema dell'approssimazione di un tensore seguendo come testo di riferimento [6].

Dato un tensore $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ di ordine N, trovare la sua migliore approssimazione di rango *mode-n* $(r_1, \ldots, r_N)$ corrrisponde a cercare un tensore $\overline{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ tale che $rank_1(\overline{\mathcal{X}}) = r_1, \ldots, rank_N(\overline{\mathcal{X}}) = r_N$ e minimizzi la seguente funzione

$$f(\overline{\mathcal{X}}) = \left\|\mathcal{X} - \overline{\mathcal{X}}\right\|^2. \tag{3.11}$$

Dire che $\overline{\mathcal{X}}$ è un tensore di rango *mode-n* $(r_1, \ldots, r_N)$ significa che può essere scritto come

$$\overline{\mathcal{X}} = \mathcal{G} \times_1 \boldsymbol{Q}^{(1)} \times_2 \boldsymbol{Q}^{(2)} \times_3 \cdots \times_N \boldsymbol{Q}^{(N)}, \tag{3.12}$$

dove $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ e $\boldsymbol{Q}^{(1)} \in \mathbb{R}^{I_1 \times R_1}, \ldots, \boldsymbol{Q}^{(N)} \in \mathbb{R}^{I_N \times R_N}$, ciascuna con colonne ortogonali tra loro.

Per minimizzare la funzione (3.11) è sufficiente determinare le matrici con colonne ortogonali $\boldsymbol{Q}^{(1)}, \ldots, \boldsymbol{Q}^{(N)}$, poiché successivamente il tensore $\mathcal{G}$ è determinato dal seguente Teorema.

Teorema 3.3. *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ e fissiamo $\boldsymbol{Q}^{(1)} \in \mathbb{R}^{I_1 \times r_1}, \ldots, \boldsymbol{Q}^{(N)} \in \mathbb{R}^{I_N \times r_N}$ matrici con colonne ortogonali. Allora il tensore $\mathcal{G}$ che minimizza $f(\overline{\mathcal{X}}) = \left\|\mathcal{X} - \overline{\mathcal{X}}\right\|^2$, con $\overline{\mathcal{X}}$ scritto come nell'equazione* (3.12)*, è il seguente:*

$$\mathcal{G} = \mathcal{X} \times_1 \left(\boldsymbol{Q}^{(1)}\right)^{\mathsf{T}} \times_2 \left(\boldsymbol{Q}^{(2)}\right)^{\mathsf{T}} \times_3 \cdots \times_N \left(\boldsymbol{Q}^{(N)}\right)^{\mathsf{T}}.$$

Di conseguenza il problema di minimizzazione dell'equazione (3.11) può essere riformulato come segue.

Teorema 3.4. *Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore N dimensionale; il problema di minimizzare l'equazione* (3.11) *è equivalente a massimizzare, attraverso matrici con colonne ortogonali, la seguente funzione:*

$$g(\boldsymbol{Q}^{(1)}, \ldots, \boldsymbol{Q}^{(N)}) = \left\|\mathcal{X} \times_1 \left(\boldsymbol{Q}^{(1)}\right)^{\mathsf{T}} \times_2 \left(\boldsymbol{Q}^{(2)}\right)^{\mathsf{T}} \times_3 \cdots \times_N \left(\boldsymbol{Q}^{(N)}\right)^{\mathsf{T}}\right\|^2. \tag{3.13}$$

Inoltre, se il tensore $\mathcal{G}$ è scelto come nel Teorema 3.3, la sua norma di Frobenius è uguale a quella di $\overline{\mathcal{X}}$, preso come nell'equazione (3.12)*, e le funzioni* (3.11) *e* (3.13) *soddisfano la seguente relazione:*

$$f = \|\mathcal{X}\|^2 - g.$$

Dimostrazione. Per prima cosa, possiamo esprimere la funzione f, nel seguente modo:

$$f(\overline{\mathcal{X}}) = \left\|\mathcal{X} - \overline{\mathcal{X}}\right\|^2 = \|\mathcal{X}\|^2 - 2\left\langle \mathcal{X}, \overline{\mathcal{X}} \right\rangle + \left\|\overline{\mathcal{X}}\right\|^2$$

Dalla definizione di prodotto interno segue che

$$\begin{aligned}
\left\langle \mathcal{X}, \overline{\mathcal{X}} \right\rangle &= \left\langle \mathcal{X}, \mathcal{G} \times_1 \boldsymbol{Q}^{(1)} \times_2 \boldsymbol{Q}^{(2)} \times_3 \cdots \times_N \boldsymbol{Q}^{(N)} \right\rangle \\
&= \left\langle \mathcal{X} \times_1 \left(\boldsymbol{Q}^{(1)}\right)^{\mathsf{T}} \times_2 \left(\boldsymbol{Q}^{(2)}\right)^{\mathsf{T}} \times_3 \cdots \times_N \left(\boldsymbol{Q}^{(N)}\right)^{\mathsf{T}}, \mathcal{G} \right\rangle \\
&= \|\mathcal{B}\|^2.
\end{aligned}$$

Siccome $\boldsymbol{Q}^{(1)}, \ldots, \boldsymbol{Q}^{(N)}$ hanno colonne ortogonali, non influenzano la norma di Frobenius; segue banalmente

$$\left\|\overline{\mathcal{X}}\right\|^2 = \|\mathcal{G}\|^2.$$

Andando a sostituire nell'espressione di f otteniamo:

$$f(\overline{\mathcal{X}}) = \|\mathcal{X}\|^2 - \|\mathcal{G}\|^2.$$

Utilizzando la definizione di g la dimostrazione è conclusa. □

Riassumendo, per trovare la migliore approssimazione di un certo rango *mode-n* di un tensore $\mathcal{X}$ bisogna trovare le matrici $\boldsymbol{Q}^{(i)}$ delle giuste dimensioni con colonne ortogonali che massimizzano $\left\|\mathcal{X} \times_1 \left(\boldsymbol{Q}^{(1)}\right)^{\mathsf{T}} \times_2 \left(\boldsymbol{Q}^{(2)}\right)^{\mathsf{T}} \times_3 \cdots \times_N \left(\boldsymbol{Q}^{(N)}\right)^{\mathsf{T}}\right\|^2$. Per questo motivo l'Algoritmo 1 determina una decomposizione di Tucker troncata che non corrisponde alla migliore approssimazione; essa però fornisce un'ottima base di partenza per un algoritmo ALS, che prende il nome di HOOI, High Order Orthogonal Iteration, e converge alla soluzione cercata. Sia $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ un tensore di rango *mode-n* $(R_1, R_2, \ldots, R_N)$, vogliamo calcolare la Tucker troncata di $\mathcal{X}$, di rango $(r_1, \ldots, r_N)$ con $0 < r_k \leq R_k$ per $k = 1, \ldots, N$:

Algorithm 2 HOOI

function HOOI($\mathcal{X}, r_1, \ldots, r_N$)

inizializzare $\boldsymbol{U}_n \in \mathbb{R}^{r_n \times I_n}$ per $n = 1, \ldots, N$ come nell'Algoritmo 1 HOSVD

repeat

for $n = 1, \ldots, N$ **do**

$\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{U}_1^{\mathsf{T}} \times_2 \cdots \times_{n-1} \boldsymbol{U}_{n-1}^{\mathsf{T}} \times_{n+1} \boldsymbol{U}_{n+1}^{\mathsf{T}} \times_{n+2} \cdots \times_N \boldsymbol{U}_N^{\mathsf{T}}$

$\boldsymbol{U}_n \leftarrow$ i primi r_n vettori singolari sinistri della SVD di $\boldsymbol{G}_{(n)}$

end for

until abbiamo una buona approssimazione o viene raggiunto un numero massimo di iterazioni

$\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{U}_1^{\mathsf{T}} \times_2 \cdots \times_N \boldsymbol{U}_N^{\mathsf{T}}$

return $\mathcal{G}, \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N$

Capitolo 4

Riconoscimento di Immagini

Il riconoscimento delle immagini, noto anche come riconoscimento di pattern, è una tecnica che, come scrivono R.O. Duda, P.E. Hart e D.G. Storck in *Pattern Classification* [7], "agisce prendendo grandi colonne di dati e compie azioni con lo scopo di classificare pattern". Al giorno d'oggi vi sono innumerevoli applicazioni: riconoscimento facciale, riconoscimento delle impronte digitali, la visione artificiale, ...

Nei vari ambiti sopracitati le "grandi colonne di dati" della definizione corrispondono a delle immagini. È infatti possibili digitalizzare un'immagine, passaggio obbligato per renderla leggibile dalla macchina, trasformando una foto reale in una corrispondente rappresentazione numerica, salvata in una matrice. Un'immagine digitalizzata in una matrice $\boldsymbol{Z} \in \mathbb{R}^{I \times J}$ può a sua volta essere riscritta in un vettore $\boldsymbol{z} \in \mathbb{R}^{IJ}$ tramite l'operazione Vec, descritta nella Sezione 1.1. Di qui a seguire il termine "immagine" viene utilizzato per indicarne la matrice numerica, o il vettore, corrispondente.

Riassumendo, riconoscere un'immagine in modo automatico significa utilizzare algoritmi in grado di ricevere immagini digitalizzate, calcolare una distanza da pattern noti e ricavarne informazioni.

In questo capitolo viene trattata la classificazione automatica di caratteri numerici scritti a mano, considerato un problema standard nel riconoscimento di pattern, vedi Capitolo 10 in [8]. In altre parole, vengono illustrati algoritmi in grado di visualizzare un'immagine contenente un numero da 0 a 9 scritto a mano e riconoscere di che cifra si tratta. Una prima applicazione fu l'automatizzazione della lettura degli indirizzi scritti sulle migliaia di lettere che arrivavano tutti i giorni alle poste americane. Questo tipo di processo è un problema difficile da risolvere poiché ciascun numero può essere scritto in innumerevoli modi differenti (ogni persona ha una propria grafia) e cifre diverse possiedono caratteristiche grafiche in comune.

Gli algoritmi differenti per il riconoscimento delle immagini proposti nelle pagine di

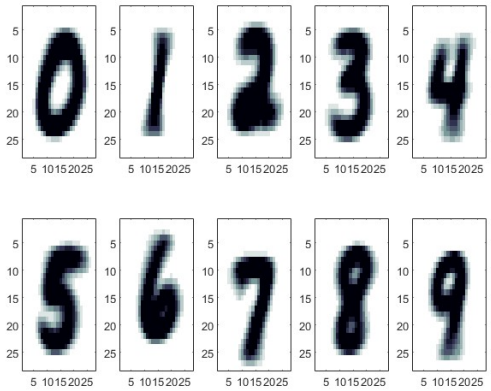

Figura 4.1: *Esempio di caratteri digitali scritti a mano del Dataset MNIST.*

seguito sono stati implementati su MATLAB e analizati sul Dataset MNIST.

MNIST è un famoso Dataset creato nel 1994, contenente circa 70000 immagini in bianco e nero di cifre scritte a mano, raccolte dall'agenzia federale americana "United States Census Bureau" e salvate in matrici 28×28. Questo Dataset è spesso utilizzato come test base per verificare la validità degli algoritmi, in quanto contiene immagini in cui le cifre sono centrali, senza rotazioni e ben scritte (facilmente distinguibili). Inoltre bisogna aggiungere che MNIST è a sua volta diviso in Training Set e Test Set, il primo contiene circa 60000 immagini ed è utilizzato per allenare l'algoritmo durante la fase di implementazione, mentre il secondo è composto da 10000 foto utilizzate per analizzare la percentuale di buon riconoscimento del metodo.

4.1 Un Algoritmo basato sui Centroidi

Quello trattato in questa sezione è un primo ed elementare algoritmo di riconoscimento di immagini basato sul calcolo delle medie, che prendono il nome di centroidi, di ciascuna cifra di tutti i caratteri scritti a mano.

Prima di illustrare l'algoritmo vero e proprio è necessario soffermarsi ulteriormente sulla struttura del Dataset MNIST, in particolare del Training Set. Ogni immagine $\boldsymbol{Z} \in \mathbb{R}^{28\times 28}$ è stata trasformata in un vettore $\boldsymbol{z} \in \mathbb{R}^{784}$ tale che $\boldsymbol{z} = Vec(\boldsymbol{Z})$; i vettori sono stati suddivisi in dieci Training Set differenti, ciascuno contenente soltanto immagini di una precisa cifra scritta da diverse persone. Notazionalmente indicheremo il Training Set contente solo le immagini della cifra k come $\boldsymbol{A}(k) = \left[\boldsymbol{z}_1^k, \boldsymbol{z}_2^k, \ldots, \boldsymbol{z}_{n_{(k)}}^k\right] \in \mathbb{R}^{784\times n_{(k)}}$ per $k = 0, \ldots, 9$; dove $\boldsymbol{z}_j^k$ rappresenta una qualsiasi immagine del carattere k, quindi

ciascuna colonna di $\boldsymbol{A}(k)$ è un'immagine, e $n_{(k)}$ è il numero di immagini della cifra k presenti tra le 60000 foto di Mnist dedicate al Training.

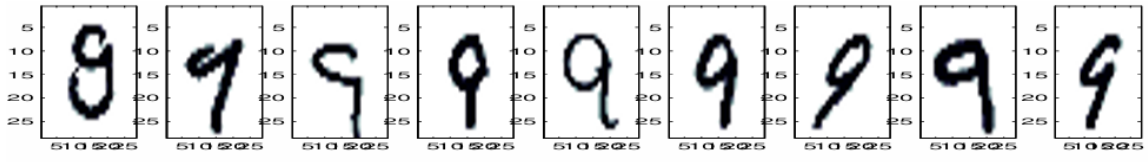

Figura 4.2: *Esempio di alcune immagini presenti in* $\boldsymbol{A}(9)$.

Coerentemente con la notazione precedente, procediamo con il caricamento dei dieci Training Set $\boldsymbol{A}(k)$ sulla macchina, e indichiamo con $\boldsymbol{z}_s \in \mathbb{R}^{784}$ un'immagine vettorializzata di una cifra della quale non conosciamo l'identità e vogliamo identificare.

Algorithm 3 Riconoscimento tramite Centroidi

% calcolo dei Centroidi $\boldsymbol{c}_k$

for $k = 0, \ldots, 9$ **do**

$\boldsymbol{c}_k \leftarrow \left(\sum_{i=1}^{n_{(k)}} \boldsymbol{z}_i^k\right) / n_{(k)}$

end for

% riconoscimento di $\boldsymbol{z}_s$

function RicCentroidi($\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_9, \boldsymbol{z}_s$)

for $k = 0, \ldots, 9$ **do**

$d_k \leftarrow \|\boldsymbol{z}_s - \boldsymbol{c}_k\|$

end for

$[\sim, Ind] \leftarrow \min([d_0, \ldots, d_9])$

return $Ind - 1$

Come si nota facilmente, l'Algortimo 3 è diviso in due parti. Nella prima parte avviene il calcolo del centroide di ogni cifra, che può essere interpretato come un'approssimazione del modo in cui la maggior parte delle persone scrive quel numero, e corrisponde alla media aritmetica di tutte le immagini presenti nel Training Set di un particolare carattere. Nella seconda parte del codice avviene il riconoscimento vero e proprio dell'immagine, che consiste nel calcolare la distanza euclidea tra l'immagine sconosciuta $\boldsymbol{z}_s$ e tutti i centroidi, e nell'identificazione di $\boldsymbol{z}_s$ con la cifra corrispondente al centroide più vicino. L'algoritmo è di facile e veloce implementazione.

Utilizzando le immagini presenti nel Test Set di MNIST, circa 1000 per cifra, è stata analizzata la validità dell'algoritmo. Al metodo con i centroidi è stato chiesto di riconoscere le immagini, delle quali sapevamo la cifra rappresentata, e confrontando il numero

proposto dall'algoritmo con il reale valore datogli in input è stata calcolata la percentuale di immagini ben identificate per ogni cifra, utilizzando la seguente formula;

$$\%\text{imm. di } k \text{ ben riconosciute} = \frac{(\#\text{imm. di } k \text{ ben allocate})}{(\#\text{imm. di } k \text{ da riconoscere})} * 100 \text{ per } k = 0, \ldots, 9. \tag{4.1}$$

È stata costruita la seguente tabella:

0	1	2	3	4	5	6	7	8	9
89.592	96.211	75.678	80.594	82.587	68.61	86.326	83.268	73.717	80.674

Tabella 4.1: Sono raffigurate le cifre e la rispettiva percentuale di immagini ben identificate.

4.2 Un Algoritmo basato sulla SVD

Il secondo algoritmo proposto in questo capitolo è basato sulla SVD matriciale. La decomposizione e la notazione del Dataset MNIST sono le stesse utilizzate nella Sezione 4.1.

L'idea alla base dell'algoritmo è quella di approssimare le immagini di ciascun Training Set tramite i vettori singolari sinistri. Fissiamo una cifra $k \in \{0, \ldots, 9\}$ e consideriamo il Dataset corrispondente $\boldsymbol{A}(k) \in \mathbb{R}^{784 \times n_{(k)}}$; la sua SVD può essere scritta come

$$\boldsymbol{A}(k) = \sum_{i=1}^{q} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}} = \sum_{i=1}^{q} \left[\sigma_i v_{1,i} \boldsymbol{u}_i, \cdots, \sigma_i v_{n,i} \boldsymbol{u}_i\right], \quad \text{con} \quad q = \min\left\{784, n_{(k)}\right\}.$$

Ogni colonna di $\boldsymbol{A}(k)$ può essere scritta come combinazione lineare dei vari vettori singolari sinistri $\boldsymbol{u}_i$, ricordando che le colonne $\boldsymbol{z}_j^k$ di $\boldsymbol{A}(k)$ sono delle immagini della cifra k, è semplice intuire che ogni vettore singolare sinistro $\boldsymbol{u}_i$ rappresenta un'immagine. Inoltre, nella scrittura di ciascuna $\boldsymbol{z}_j^k$ il coefficiente di $\boldsymbol{u}_i$ corrisponde a $\sigma_i v_{j,i}$, poiché $\sigma_1 \geq \cdots \geq \sigma_q > 0$, si può affermare che i vari $\boldsymbol{u}_i$ rappresentano immagini via via meno dominanti.

Esistono alcune ipotesi preliminari che garantiscono il buon funzionamento dell'algoritmo. (1) Ogni numero può essere ben rappresenatato da pochi vettori singolari. (2) La distinzione tra due cifre diverse può avvenire con l'utilizzo di pochi vettori singolari. (3) Se la distanza tra un'immagine e un certo carattere del Training Set è la più piccola, la probabilità che siano due numeri diversi deve essere bassa (bassa percentuale di errore). Le immagini presenti in MNIST soddisfano queste caratteristiche.

Sia $\boldsymbol{z}_s \in \mathbb{R}^{784}$ un'immagine da classificare e sia $k \in \{0, \ldots, 9\}$ una cifra, consideriamo il Training Set corrispondente $\boldsymbol{A}(k) \in \mathbb{R}^{784 \times n_{(k)}}$ e supponiamo che esso possa essere ben rappresentato soltanto dai primi J vettori singolari,

$$\boldsymbol{A}(k) \approx \sum_{i=1}^{J} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}} = \boldsymbol{U}_J \boldsymbol{\Sigma}_J \boldsymbol{V}^{\mathsf{T}}, \quad \text{con} \quad \boldsymbol{U}_J = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J].$$

In realtà $J = J_{(n)}$, infatti il valore di troncamento può essere diverso per ogni Training Set $\boldsymbol{A}(k)$, e ovviamente $\boldsymbol{U}_J = \boldsymbol{U}_J^k$ in quanto sono i vettori singolari relativi alla cifra k; la dipendenza da k è stata omessa per alleggerire la notazione. La distanza tra $\boldsymbol{z}_s$ e $\boldsymbol{A}(k)$, detta anche residuo, è definita come segue,

$$dist\left(\boldsymbol{z}_s, \boldsymbol{A}(k)\right) = \min_{\boldsymbol{\alpha} \in \mathbb{R}^J} \left\| \boldsymbol{z}_s - \sum_{i=1}^{J} \alpha_i \boldsymbol{u}_i \right\| \equiv \left\| \boldsymbol{z}_s - \boldsymbol{U}_J \boldsymbol{U}_J^{\mathsf{T}} \boldsymbol{z}_s \right\|;$$

e coincide con la proiezione di $\boldsymbol{z}_s$ nello spazio ortogonale a $Range(\boldsymbol{U}_J)$. Per facilitare la lettura dell'algoritmo, solo nella computazione, $\boldsymbol{U}_J$ verrà indicata come $\boldsymbol{U}_J^k$.

Algorithm 4 Riconoscimento tramite SVD

% calcolo delle SVD

for $k = 0, \ldots, 9$ **do**

$\boldsymbol{U}_J^k \leftarrow$ i primi J vettori singolari sinistri della SVD di $\boldsymbol{A}(k)$

end for

% riconoscimento di $\boldsymbol{z}_s$

function RicSVD($\boldsymbol{U}_J^0, \boldsymbol{U}_J^1, \ldots, \boldsymbol{U}_J^9, \boldsymbol{z}_s$)

for $k = 0, \ldots, 9$ **do**

$res_k \leftarrow \left\| \boldsymbol{z}_s - \boldsymbol{U}_J^k \left(\boldsymbol{U}_J^k\right)^{\mathsf{T}} \boldsymbol{z}_s \right\|$

end for

$[\sim, Ind] \leftarrow \min([res_0, \ldots, res_9])$

return $Ind - 1$

Nella prima parte dell'algoritmo avviene il calcolo della SVD dei dieci Training Set $\boldsymbol{A}(k)$,. È importante notare che il metodo richiede soltanto il calcolo delle prime J triplette delle varie SVD e non della decomposizione completa, molto più costosa. Nella funzione di riconoscimento, l'immagine viene semplicemente classificata come la cifra corrispondente al residuo minore.

L'inizializzazione dell'Algoritmo 4 richiede la scelta del parametro di troncamento J. Per verificare quanto tale scelta influenza le prestazioni dell'algoritmo, per ogni $J = 1, \ldots, 20$, utilizzando le circa 1000 immagini per cifra presenti nel Test Set di MNIST, è

stata calcolata la percentuale totale di numeri ben riconosciuti,

$$\%\text{imm. totali ben riconosciute} = \frac{(\#\text{imm. totali ben allocate})}{(\#\text{imm. totali da riconoscere})} * 100. \tag{4.2}$$

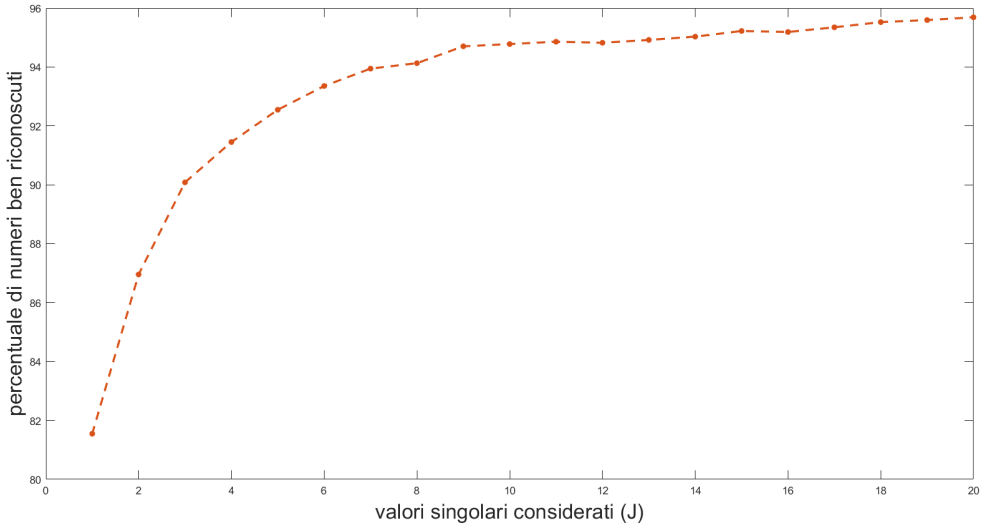

Figura 4.3: *Grafico che rappresenta il variare della percentuale di immagini ben riconosciute (sulle y) all'aumentare di* $J = 1, \ldots, 20$ *(sulle x).*

Come si osserva dalla Figura 4.3, l'aumento del numero J di vettori singolari considerati porta ad un significativo miglioramento delle prestazione dell'algoritmo passando dall'82% per $J = 1$ a circa il 96% per $J = 20$. Però, dopo la veloce crescita iniziale della percentuale di buon riconoscimento, si nota che quest'ultima tende a stabilizzarsi, di conseguenza, la scelta di un J grande, non soltanto non porta a un effettivo miglioramento dell'algoritmo ma ne aumenta anche il costo computazionale, dovendo calcolare più triplette dell'SVD.

Fissato un valore di J e utilizzando l'equazione (4.1), è stata creata la seguente tabella:

0	1	2	3	4	5	6	7	8	9
98.776	99.471	93.314	93.168	97.454	93.498	96.973	92.899	92.71	93.954

Tabella 4.2: Sono raffigurate le cifre e la rispettiva percentuale di immagini ben identificate, con $J = 15$ per ciascun numero.

Le percentuali di buona allocazione sono decisamente superiori rispetto ai risultati della Tabella 4.1 e sottolineano la migliore precisione dell'algoritmo basato sull'SVD rispetto a quello dei centroidi, fatto che giustifica il maggior costo computazionale di questo secondo metodo.

4.3 Riconoscimento con HOSVD

Il terzo e ultimo algoritmo proposto in questo capitolo è un esempio di applicazione dei tensori e della decomposizione HOSVD; infatti, vista la loro struttura mutidimensionale, i tensori si prestano ad essere utilizzati nell'analisi di dati in cui si hanno più variabili contemporaneamente.

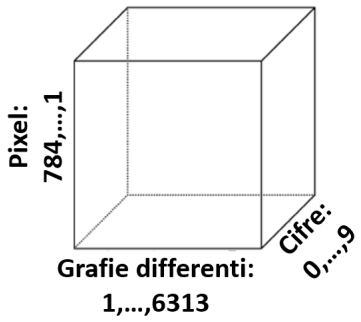

Figura 4.4: *Rappresentazione del Test Set di MNIST strutturato come un tensore* $784 \times 6313 \times 10$. *Per "Grafie differenti" si intendono immagini della stessa cifra scritte da persone diverse.*

Prima di procedere con l'esposizione del metodo di riconoscimento tramite HOSVD è necessario spiegare come il Training Set di MNIST è stato strutturato in un tensore. Il tensore $\mathcal{X}$ cubico del Training Set è stato creato con i seguenti criteri: le 1-fibre sono immagini, lungo il terzo modo sono disposte le dieci differenti cifre e le facce frontali corrispondono con i Training Set dei vari caratteri, ovvero,

$$\mathcal{X}(:,:,k+1) = \boldsymbol{A}(k), \quad \text{per} \quad k = 0, \ldots, 9.$$

Poiché non tutti i Training Set $\boldsymbol{A}(k)$ hanno lo stesso numero $n_{(k)}$ di immagini, il tensore è stato creato uniformando la quantità di foto per ogni cifra. Quello ottenuto è un tensore $\mathcal{X} \in \mathbb{R}^{784\times6313\times10}$; la lunghezza del primo modo viene indicata come n_i e corrisponde alla dimensione di ciascuna immagine vettorializzata (numero di pixel), la lunghezza del secondo modo viene indicato come n_e e corrisponde al numero di immagini della stessa cifra scritta in modo diverso (numero di espressioni), infine la lunghezza del terzo modo

è indicata come n_p e corrisponde al numero di cifre, $\mathcal{X} \in \mathbb{R}^{n_i \times n_e \times n_p}$. Notazionalmente indicheremo gli unfolding lungo i vari modi di $\mathcal{X}$ come $\boldsymbol{X}_{(i)} = \boldsymbol{X}_{(1)}$, $\boldsymbol{X}_{(e)} = \boldsymbol{X}_{(3)}$ e $\boldsymbol{X}_{(p)} = \boldsymbol{X}_{(3)}$.

Sia $\mathcal{X} \in \mathbb{R}^{n_i \times n_e \times n_p}$ il tensore del Test Set, $\boldsymbol{z}_s \in \mathbb{R}^{n_i}$ un'immagine da riconoscere e $m_i\,(\leq n_i)$, $m_e\,(\leq n_e)$ e $m_p\,(\leq n_p$, solitamente si prendono uguali) i tre valori di troncamento utilizzati nei rispettivi modi ai fini del calcolo dell'HOSVD troncata.

Algorithm 5 Riconoscimento tramite HOSVD troncata

% calcolo delle matrici della HOSVD

$[\mathcal{S}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}] \leftarrow HOSVD(\mathcal{X}, m_i, m_e, m_p)$ % matrici della decomposizione troncata ottenute richiamando l'Algoritmo 1

$\mathcal{C} \leftarrow \mathcal{S} \times_1 \boldsymbol{F} \times_2 \boldsymbol{G}$

% riconoscimento di $\boldsymbol{z}_s$

function RicHOSVD($\mathcal{C}, \boldsymbol{H}, n_e, n_p, \boldsymbol{z}_s$)

for $e = 1, \ldots, n_e$ **do**

$\boldsymbol{C}^e = \mathcal{C}(:, e, :)$ % spazio dell'espressione e

$\hat{\boldsymbol{\alpha}}_e = \min_{\boldsymbol{\alpha}_e} \{\|\boldsymbol{C}^e \boldsymbol{\alpha}_e - \boldsymbol{z}_s\|_2\} = \boldsymbol{C}^e \backslash \boldsymbol{z}_s$ % coordinate di $\boldsymbol{z}_s$ in tale spazio

for $p = 1, \ldots, n_p$ **do**

$\boldsymbol{h}_p \leftarrow \boldsymbol{H}(p, :)$ % coordinata di ogni p nello spazio $\boldsymbol{C}^e$

$D(e, p) \leftarrow \left\|\hat{\boldsymbol{\alpha}}_e - \boldsymbol{h}_p^{\mathsf{T}}\right\|_2$ % matrice delle distanze

end for

end for

$\boldsymbol{d1} \leftarrow \min_e\{\boldsymbol{D}\}$ % distanza di $\boldsymbol{z}_s$ dall'espressione più vicina di ogni cifra

$[\sim, Ind] \leftarrow \min_p\{\boldsymbol{d1}\}$

return $Ind - 1$

Come prima cosa, grazie all'Algoritmo 1, si devono calcolare i fattori della decompisione HOSVD troncata del tensore $\mathcal{X} \in \mathbb{R}^{n_i \times n_e \times n_p}$, che può essere scritta come:

$$\mathcal{X} = \mathcal{S} \times_1 \boldsymbol{F} \times_2 \boldsymbol{G} \times_3 \boldsymbol{H} = \mathcal{C} \times_3 \boldsymbol{H}, \quad \text{con} \quad \mathcal{C} = \mathcal{S} \times_1 \boldsymbol{F} \times_2 \boldsymbol{G}.$$

Dalla definizione di prodotto *mode-n* vale,

$$\mathcal{X}(:, e, :) = \mathcal{C}(:, e, :)\boldsymbol{H}^{\mathsf{T}} \quad \text{per} \quad e = 1, \ldots, n_e.$$

Per la struttura data al tensore, l'immagine della cifra p con l'espressione e, indicata come $\boldsymbol{x}_p^{(e)}$, si trova in posizione $\mathcal{X}(:, e, p)$, è quindi possibile scrivere,

$$\boldsymbol{x}_p^{(e)} = \mathcal{X}(:, e, p) = \boldsymbol{C}^e \left(\boldsymbol{h}_p\right)^{\mathsf{T}} \quad \text{con } \boldsymbol{C}^e = \mathcal{C}(:, e, :) \in \mathbb{R}^{n_i \times n_p} \text{ e } \boldsymbol{h}_p = \boldsymbol{H}(p, :) \in \mathbb{R}^{1 \times n_p}.$$

L'uguaglianza precedente può essere interpretata geometricamente come segue: denotiamo con E^e lo spazio vettoriale dell'espressione e (l'insieme delle immagini di cifre diverse scritte dalla stessa persona), allora le colonne di $\boldsymbol{C}^e$ non sono altro che una base di E^e, mentre le righe di $\boldsymbol{H}$, ovvero $\boldsymbol{h}_p$, corrispondono alle coordinate della cifra p in tale base.

Per trovare la distanza tra l'immagine da riconoscere $\boldsymbol{z}_s$ e una cifra p, per prima cosa si calcolano le coordinate $\hat{\boldsymbol{\alpha}}_e$ di $\boldsymbol{z}_s$ negli spazi vettoriali E^e,

$$\hat{\boldsymbol{\alpha}}_e = \arg\min_{\boldsymbol{\alpha}_e} \{\|\boldsymbol{C}^e\boldsymbol{\alpha}_e - \boldsymbol{z}_s\|_2\} \quad \text{per } e = 1, \ldots, n_e;$$

dopo di che si cerca la distanza minore tra le coordinate di $\boldsymbol{z}_s$ nelle varie espressioni e quelle di p,

$$dist(\boldsymbol{z}_s, p) = \min_{e=1,\ldots,n_e} \left\{ \left\| \hat{\boldsymbol{\alpha}}_e - \boldsymbol{h}_p^{\mathsf{T}} \right\|_2 \right\}.$$

L'immagine $\boldsymbol{z}_s$ viene identificata con la cifra p alla quale è più vicina,

$$\boldsymbol{z}_s = \arg\min_{p=1,\ldots,n_p} \{dist(\boldsymbol{z}_s, p)\}.$$

Nell'algoritmo viene costruita la matrice delle distanze $\boldsymbol{D}$, che contiene in posizione (e, p) la distanza tra $\boldsymbol{z}_s$ e p nello spazio E^e.

L'inizializzazione dell'Algoritmo 5 richiede la scelta di due parametri di troncamento dell'HOSVD, m_i e m_e ($m_p = n_p$). La scelta di questi parametri non è fondamentale soltanto per rendere l'algoritmo efficiente, ma anche affinché il conto sia computazionalmente affrontabile dalla macchina. Infatti, lavorando con un tensore $\mathcal{X} \in \mathbb{R}^{784\times 6313\times 10}$ di grandi dimensioni, il calcolo della HOSVD completa sarebbe molto costoso. Il troncamento m_i agisce sul modo dei pixel delle immagini; intuitivamente, avere un alto valore di m_i significa lavorare con immagini nitide mentre avere un valore basso significa utilizzare immagine più sgranate, quindi più difficili da riconoscere. Analogamente, m_e approssima lungo il modo delle espressioni; avere un alto valore m_e significa allenare l'algoritmo con cifre scritte da tante persone diverse, e la varietà di grafia aiuta nel riconoscimento. Dall'altra parte però avere tante immagini molto simili dello stessa cifra non aggiunge nessuna informazione e appesantisce il metodo. La ricerca dei migliori troncamenti è stata affrontata in due modi diversi.

Il primo metodo utilizzato per cercare i migliori valori di troncamento visualizza gli autovalori delle SVD dei vari unfolding al fine di scegliere graficamente il numero massimo m_i (o m_e) di autovettori sinistri da considerare, ricordando che gli autovettori dominanti sono in corrispondenza degli autovalori maggiori.

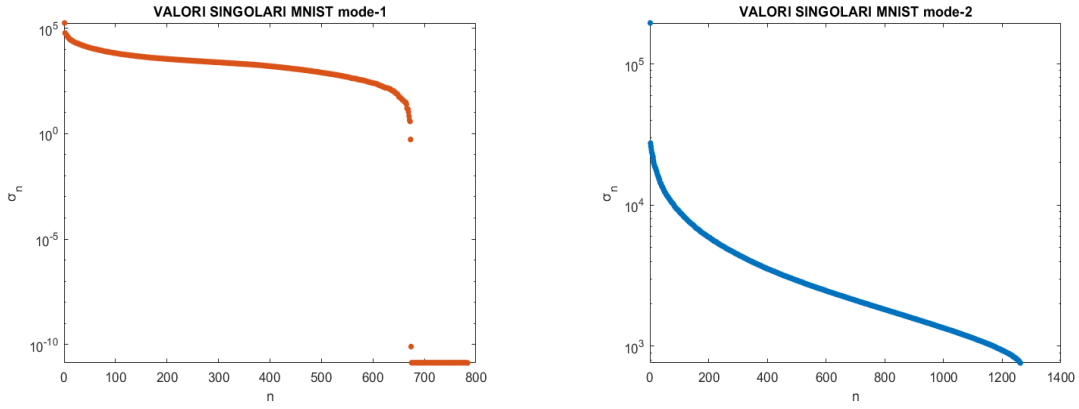

Figura 4.5: *Stampa dei valori singolari rispettivamente di* $\boldsymbol{X}_{(i)}$ *(Pixel, in arancione) e* $\boldsymbol{X}_{(e)}$ *(Espressioni, in blu) del Training Set*

Come test preliminare, sono stati identificati $m_i(max) = 40$ e $m_e(max) = 65$ come valori in cui la curva degli autovalori inizia a scendere. Una volta fissati tali parametri, l'algoritmo è stato testato più volte, facendo variare simultaneamente $m_i = 1, \ldots, m_i(max)$ e $m_e = 1, \ldots, m_e(max)$, calcolando ogni volta la percentuale di buona allocazione totale, con la formula (4.2), e il tempo impiegato per la scomposizione e il riconoscimento, indicatore del costo computazionale dell'algoritmo. I dati ottenuti sono stati salvati in due matrici differenti, $\boldsymbol{A} \in \mathbb{R}^{m_i(max) \times m_e(max)}$ e $\boldsymbol{B} \in \mathbb{R}^{m_i(max) \times m_e(max)}$, tali che in posizione (i, e) vi sono rispettivamente la percentuale di buona allocazione e il costo computazionale (in secondi) dell'algoritmo inizializzato con $m_i = i$ e $m_e = e$.

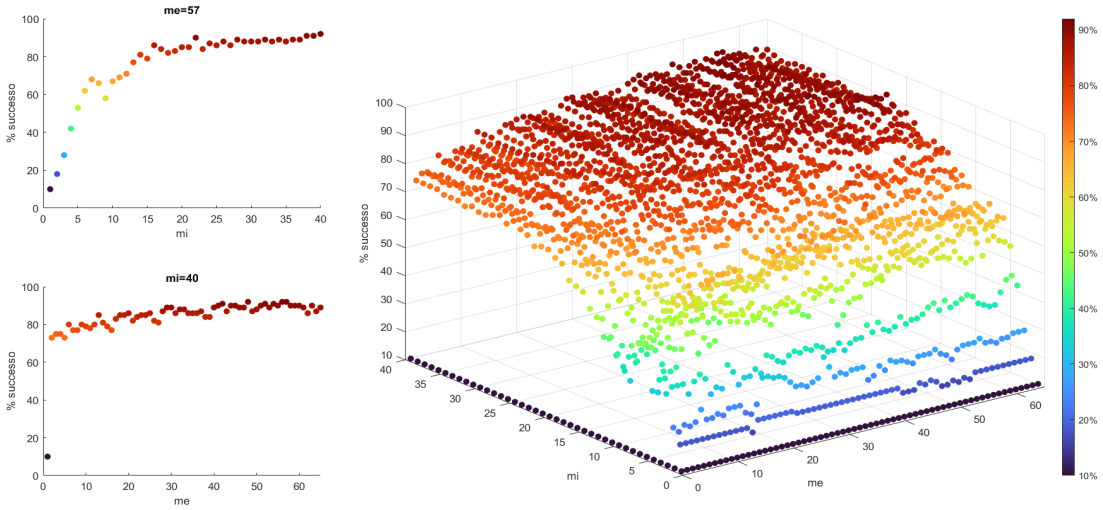

Figura 4.6: *Rappresentazione tridimensionale della percentuale di buona allocazione (sulle z), detta anche % di successo, al variare di* m_e *(sulle x) e* m_i *(sulle y).*

La Figura 4.6, che di fatto è la stampa dei punti $(e, i, \boldsymbol{A}(i, e))$, conferma che aumentando entrambi i parametri la percentuale di buona allocazione aumenta, essa però mostra anche che le prestazioni dell'algoritmo non crescono in modo "simmetrico"; ovvero, per bassi valori di m_e si ottengono comunque valori di successo accettabili mentre per bassi valori di m_i l'algoritmo fatica a riconoscere le immagini. Questo fatto può essere tradotto affermando che, piuttosto che lavorare con tante immagini di qualità pessima, è meglio utilizzare meno immagini ma di qualità maggiore.

Questa prima analisi qualitativa è terminata cercando i valori dei due troncamenti che meglio garantissero ottime prestazioni dell'agoritmo insieme ad un costo computazionale moderato, risolvendo,

$$\arg\min_{m_i, m_e} \left\{ \left\| 1 - \%buon\_riconoscimento(m_i, m_e) \right\|_2 + \lambda \left\| costo\_comp(m_i, m_e) \right\|_2 \right\}. \quad (4.3)$$

Siano $\boldsymbol{A} \in \mathbb{R}^{m_i(max) \times m_e(max)}$ e $\boldsymbol{B} \in \mathbb{R}^{m_i(max) \times m_e(max)}$ le matrici definite precedentemente che contengono rispettivamente le percentuali di buona allocazione e il costo computazionale dell'algoritmo al variare dei troncamenti, allora l'equazione (4.3) può essere risolta sperimentalmente cercando le coordinate del valore minimo della matrice

$$(\mathbb{1}_{m_i(max)} \mathbb{1}_{m_e(max)}^{\mathsf{T}} - \boldsymbol{A}) + \lambda \boldsymbol{B} \in \mathbb{R}^{m_i(max) \times m_e(max)},$$

dove $\mathbb{1}_{m_i(max)} \in \mathbb{R}^{m_i(max)}$ e $\mathbb{1}_{m_e(max)} \in \mathbb{R}^{m_e(max)}$ sono vettori di tutti uni.

Scegliendo dei valori di λ si sono ottenuti i seguenti risultati.

λ	m_i	m_e	**costo computazionale (s)**	**% buon riconoscimento**
0.5	21	1	11.031	10
0.05	25	8	17.984	80
0.005	40	57	21.359	92
0.0005	40	57	21.359	92

Tabella 4.3: Raffigurazione dei migliori parametri m_i e m_e utilizzando l'equazione (4.3)

Nell'equazione (4.3) λ svolge il ruolo di "peso" del costo computazionale, per un alto valore di λ si ottengono valori che corrispondono ad un algoritmo poco costoso ma altrettanto poco preciso, per un valore più basso di λ si ottengo valori di troncamento che implicano alta prestazione ma alto costo. Tutto questo è confermato dalla Tabella 4.3, che inoltre sottolinea nuovamente l'importanza di avere immagini di alta qualità in quanto gli m_i trovati corrispondo ad $m_i(max) = 40$.

Nel secondo metodo utilizzato per cercare i migliori valori di troncamento si approssimano con più precisione gli unfolding $\boldsymbol{X}_{(i)}$ e $\boldsymbol{X}_{(e)}$. Infatti, scelta una certa tolleranza

(ad esempio $tol = 10^{-2}$), i troncamenti sono presi secondo le seguenti equazioni,

$$m_i = \min_{J\in\{1,\cdots,n_i\}} \left\{ \left\| \boldsymbol{X}_{(i)} - \boldsymbol{X}_{(i)}^{(J)} \right\|_2 < tol \left\| \boldsymbol{X}_{(i)} \right\| \right\};$$
$$m_e = \min_{J\in\{1,\cdots,n_e\}} \left\{ \left\| \boldsymbol{X}_{(e)} - \boldsymbol{X}_{(e)}^{(J)} \right\|_2 < tol \left\| \boldsymbol{X}_{(e)} \right\| \right\}; \tag{4.4}$$

dove $\boldsymbol{X}_{(i)}^{(J)}$ e $\boldsymbol{X}_{(e)}^{(J)}$ sono le approssimazioni rispettivamente di $\boldsymbol{X}_{(i)}$ e $\boldsymbol{X}_{(e)}$ ottenute considerando soltanto le prime J triplette delle rispettive SVD. L'implementazione di tale criterio sfrutta il seguente risultato, per il quale la sua implementazione si riduce al calcolo degli autovalori di $\boldsymbol{X}_{(i)}$ e $\boldsymbol{X}_{(e)}$.

Proposizione 4.1. *Sia $\boldsymbol{X} \in \mathbb{R}^{I\times K}$ una matrice la cui decomposizione SVD può essere scritta come,*

$$\boldsymbol{X} = \sum_{r=1}^{R} \sigma_r \boldsymbol{u}_r \circ \boldsymbol{v}_r, \quad con \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R > 0,$$

con $R = \min\{I, K\}$. Denotiamo come $\boldsymbol{X}^J$ l'approssimazione di $\boldsymbol{X}$ ottenuta considerando soltanto le prime J triplette della decomposizione, ovvero,

$$\boldsymbol{X}^J = \sum_{j=1}^{J} \sigma_j \boldsymbol{u}_j \circ \boldsymbol{v}_j, \quad con \quad J < R.$$

Allora vale,

$$\left\| \boldsymbol{X} - \boldsymbol{X}^J \right\|_2 = \sigma_{J+1}.$$

Questa scelta dei parametri garantisce un'algoritmo con un'elevata percentuale di buon riconoscimento ma trascura il costo computazionale. Al fine di ovviare a questo problema e poter applicare l'Algoritmo 5 con dei troncamenti ottimali, sono stati eseguiti tre differenti passaggi. (1) Determinazione dei due valori di troncamento che risolvono l'equazione (4.4) con $\mathcal{X}$ Data Set MNIST completo. (2) Applicazione dell'Algoritmo 5 al variare di m_i e m_e in un intorno dei valori di troncamento ottenuti nel punto precedente, e calcolo del costo computazionale e percentuale di buon riconoscimento (con conseguente salvataggio dei dati in matrici). (3) Applicazione dell'equazione (4.3) ai dati ottenuti nel punto (2) con λ opportuno. Seguendo questo schema si sono ottenuti valori di troncamento che hanno portato risultati più prestanti di tutti i precedenti.

m_i	m_e	**costo computazionale (s)**	**% buon riconoscimento**
268	521	31.719	98

Tabella 4.4: Raffigurazione dei migliori parametri m_i e m_e

Considerando le dimensioni del tensore del Training Set di MNIST $\mathcal{X} \in \mathbb{R}^{784\times6313\times10}$ si nota che il valore ottimale del troncamento $m_e(= 521)$ è decisamente inferiore rispetto alla varietà di immagini proposte per ogni cifra (ricordiamo che in questo caso $m_e \leq \min\{6313, 7840\}$). Questo fatto fa pensare che dopo un certo numero di immagini per carattere l'algoritmo non trae più nessuna informazione utile ai fini del riconoscimento ed è soltanto sovraccaricato di informazioni. La sovrabbondanza di informazioni è detta Oversampling e influenza negativamente il costo computazionale dell'algoritmo.

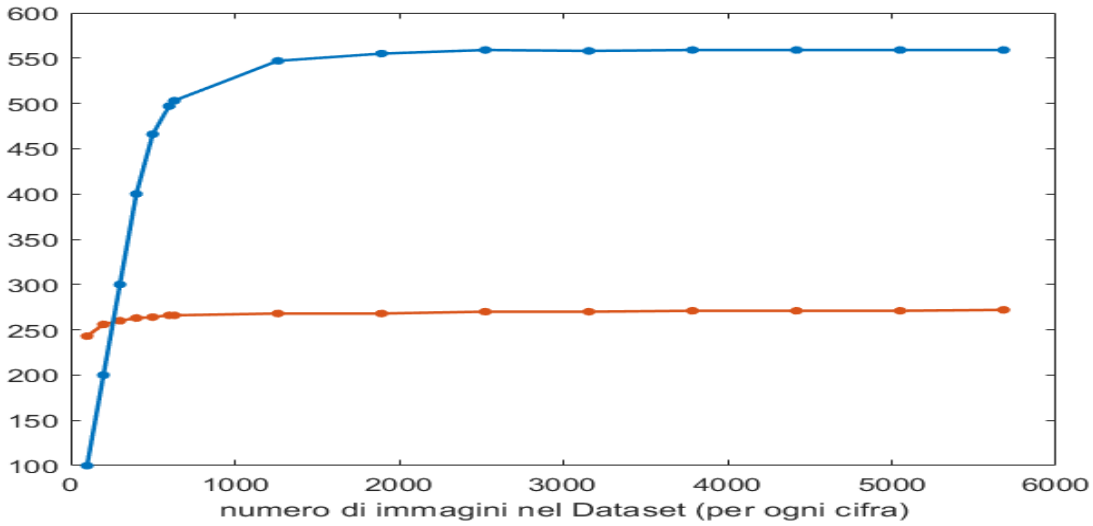

Figura 4.7: *Grafico che rappresenta il valore dei migliori m_i (in arancione) ed m_e (in azzurro) al variare del numero di immagini nel Training set per ogni cifra (secondo modo del tensore $\mathcal{X}$).*

La Figura 4.7 conferma che superate le circa 1200 immagini il valore del migliore m_e si stabilizza, di conseguenza aggiungere ulteriori immagini aumenta soltanto le dimensione del Training Set e del costo computazionale dell'algoritmo e non porta ad un miglioramento delle performance di quest'ultimo. In statistica è un noto risultato che se si lavora con un buon Training Set, che nel nostro caso significa avere immagini sufficientemente diverse e scelte in modo casuale, esso non deve necessariamente essere grande. Infatti, ai fini della percentuale di riconoscimento lavorare con 1200 immagini per ogni cifra o 6313 è indifferente, quindi per velocizzare l'algoritmo senza perdere informazioni basta lavorare con $\mathcal{X} \in \mathbb{R}^{784\times1200\times10}$. Dall'altra parte m_i è fin da subito elevato e si stabilizza velocemente, riaffermando che ai fini del riconoscimento è opportuno lavorare con immagini di alta qualità.

Bibliografia

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

[2] J. B. Kruskal, *Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics*, Linear Algebra Appl., 18 (1977)

[3] G. Golub e Ch. Van Loan, *Matrix Computations*, 4 Ed, Johns Hopkins Univ. Press., (2013).

[4] N. D. Sidiropoulos e R. Bro, *On the uniqueness of multilinear decomposition of N-way arrays*, J. Chemometrics, 14 (2000).

[5] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000).

[6] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-(R1,R2,...,RN) approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000).

[7] R. O. Duda, P. E. Hart e D. G. Storck, *Pattern Classification*, 2 Ed, (2001).

[8] Lars Eldén, *Matrix Methods in Data Mining and Pattern Recognition*, SIAM, (2007).