Alma Mater Studiorum · Università di Bologna

**SCUOLA DI SCIENZE**
**Corso di Laurea in Matematica**

# LA DECOMPOSIZIONE CUR

# PER IL SUBSPACE CLUSTERING

**Tesi di Laurea in Matematica Computazionale**

**Relatore:**
**Chiar.ma Prof.ssa**
**VALERIA SIMONCINI**

**Presentata da:**
**IRENE NANNINI**

**Anno Accademico 2023/2024**

*Alla mia famiglia...*

# Introduzione

In molte applicazioni dell'analisi dei dati, due strumenti chiave sono la riduzione dimensionale e la compressione, dove i dati, rappresentati come vettori in uno spazio euclideo di dimensione alta, vengono approssimati in una base o "frame" che genera uno spazio di dimensioni molto inferiori rispetto allo spazio ambiente. Tuttavia, è essenziale bilanciare il metodo di approssimazione con la necessità che qualsiasi risultato ottenuto dalle versioni approssimate dei dati sia facilmente interpretabile. Molti metodi ben noti, come la PCA (Principal Component Analysis), permettono una grande approssimazione e compressione dei dati, ma spesso a costo di ostacolare l'interpretazione dei risultati. Ad esempio, l'autovettore

$$[(1/2)\text{età} - (1/\sqrt{2})\text{altezza} + (1/2)\text{reddito}],$$

risulta essere uno dei "fattori" o "caratteristiche" significativi e non correlati di un dataset di caratteristiche delle persone, ma non è particolarmente informativo. Dopotutto, i vettori singolari sono astrazioni matematiche che possono essere calcolate per qualsiasi matrice di dati, e dunque difficili da interpretare. Un modo per superare questa difficoltà è tentare di utilizzare l'*auto-espressività* dei dati, ovvero la nozione secondo cui spesso i dati sono meglio rappresentati in termini di combinazioni lineari di altri punti dati piuttosto che in una base astratta. La domanda quindi è: come si può utilizzare l'auto-espressività per ottenere la riduzione dimensionale? L'idea presentata in questo elaborato è quella di scegliere le colonne e le righe più rappresentative che catturano le informazioni essenziali della matrice iniziale. Viene così introdotta la decomposizione CUR, anche conosciuta come decomposizione "skeleton", che rappresenta un metodo potente e flessibile per analizzare e approssimare matrici di grandi dimensioni.

Ci sono due approcci distinti nella maggior parte della letteratura riguardante la decomposizione CUR. Il primo è considerarla come una decomposizione o fattorizzazione esatta della matrice: in particolare, la decomposizione CUR esatta consente di esprimere una matrice $A$ come il prodotto di tre matrici, $A = CU^{\dagger}R$, dove

- $C$ è un sottoinsieme delle colonne di $A$,

- $R$ è un sottoinsieme delle righe di $A$,
- $U$ è l' "intersezione" tra $C$ e $R$.

Il secondo approccio è trovare un'approssimazione a rango basso di una matrice data: in questo contesto, l'approssimazione CUR di una matrice è tipicamente data da $A \approx CC^{\dagger}AR^{\dagger}R$, dove $CC^{\dagger}$ e $R^{\dagger}R$ sono rispettivamente proiezioni ortogonali sugli spazi generati dalle colonne e dalle righe date. La decomposizione CUR è inoltre un metodo di approssimazione a rango basso più fedele alla struttura dei dati rispetto ad altre fattorizzazioni. In questo caso infatti, $C$ e $R$ possono essere utilizzate al posto degli autovettori, e saranno più facilmente interpretabili in termini dell'ambito da cui i dati sono tratti, dato che consistono in parti di dati effettivi. Questa versatilità rende la decomposizione CUR una scelta ottima per la riduzione della dimensionalità e la gestione della complessità computazionale in vari sistemi fisici, tra cui il riconoscimento facciale e la segmentazione del moto.

Nel problema del *Subspace Clustering* (si veda la sezione 2.2), si parte da una matrice di dati tratti dall'unione di sottospazi indipendenti, e l'obiettivo è individuare una matrice di similarità che permetta di determinare il numero di sottospazi, la loro dimensione e una base per ciascuno di essi, così da poter raggruppare i dati derivanti dallo stesso sottospazio nello stesso cluster. In questo elaborato, mostreremo che la decomposizione CUR permette di costruire molte matrici di similarità per dati tratti da sottospazi, e verrà presentato anche un particolare algoritmo di clustering, il Principal Coordinate Clustering, che applicato ad una matrice di similarità restituirà per la maggior parte i clusters corretti.

Infine, ogni matrice di similarità derivante dalla decomposizione CUR può essere modificata per fare clustering su dati di rumore: utilizzare la media di molte decomposizioni CUR per creare matrici di similarità è più efficiente e veloce di molti altri metodi. Di conseguenza, presentiamo un algoritmo che permetta di fare clustering su dati di rumore, e lo testiamo sul dataset Hopkins155 per la segmentazione del moto e sul dataset Database of Faces per il riconoscimento facciale.

# Notazioni

Nel seguito, sarà spesso necessario riferirsi a sottomatrici di una matrice $A \in \mathbb{K}^{m\times n}$ formata da determinate colonne e righe. A questo scopo, verrà utilizzata la notazione nello stile di Matlab: $A(I,:)$ denoterà la sottomatrice delle righe di $A$ di dimensione $|I| \times n$, composta solo dalle righe di $A$ indicizzate da $I \subset \{1,\dots,m\}$, e allo stesso modo

$A(:, J)$ indicherà la sottomatrice delle colonne di $A$ di dimensione $m \times |J|$, composta solo dalle colonne di $A$ indicizzate da $J \subset \{1, \ldots, n\}$. Pertanto, $A(I, J)$ sarà la sottomatrice di dimensione $|I| \times |J|$ formata dagli elementi $a_{ij}$ di $A$ per i quali $(i, j) \in I \times J$. È importante notare che gli indici in $I$ e $J$ non devono essere necessariamente distinti, il che significa che possono esserci elementi ripetuti.

# Indice

# Capitolo 1

# Preliminari e notazioni

## 1.1 Richiami di Algebra Lineare

Nel seguito, il simbolo $\mathbb{K}$ sarà utilizzato per indicare il campo dei numeri reali $\mathbb{R}$ oppure quello dei numeri complessi $\mathbb{C}$. Inoltre, considerando una matrice $A \in \mathbb{K}^{m\times n}$, $\mathcal{N}(A)$ rappresenterà lo spazio nullo di $A$, definito come l'insieme $\mathcal{N}(A) = \{x \in \mathbb{K}^n : Ax = 0\}$, mentre $\mathcal{R}(A)$ indicherà il range di $A$, ossia lo spazio generato dalle sue colonne. Iniziamo introducendo alcune definizioni e risultati preliminari che verrano usati in questo elaborato.

**Definizione 1.1.** *Sia $A \in \mathbb{C}^{m\times n}$. La **trasposta coniugata** di $A$ è la matrice denotata con $A^* \in \mathbb{C}^{m\times n}$ che ha come elementi $A^*(i,j) = \overline{A(j,i)}$, dove $\overline{A(j,i)}$ è il numero complesso coniugato di $A(j,i)$.*

*Osservazione* 1.2. Nel caso reale, la trasposta coniugata di $A$ è semplicemente la trasposta $A^T$, che si ottiene scambiando tra di loro le righe e le colonne di $A$.

*Osservazione* 1.3. Per la trasposta coniugata valgono le seguenti proprietà: date $A$ e $B$ matrici con dimensioni compatibili,

- $(A+B)^* = A^* + B^*$;
- $(AB)^* = B^*A^*$;
- $(\alpha A)^* = \overline{\alpha}A^*$, per ogni $\alpha \in \mathbb{C}$.

**Definizione 1.4.** *Sia $A \in \mathbb{C}^{n\times n}$. $A$ si dice **unitaria** se vale $A^*A = AA^* = I$. Analogamente, nel caso reale $A$ si dice **ortogonale** se vale $AA^T = A^TA = I$.*

**Definizione 1.5.** *Sia $S$ un sottospazio di $\mathbb{C}^m$. Si definisce il **proiettore ortogonale** $P_S$ di $\mathbb{C}^m$ su $S$ ponendo $P_Su = u$ se $u \in S$ e $P_Su = 0$ se $u \in S^\perp$.*

*Osservazione* 1.6. Se $Q \in \mathbb{R}^{n\times k}$, con $k < n$, è una matrice con colonne ortonormali, la sua proiezione ortogonale sarà data dalla matrice $\Pi = QQ^T$, che è simmetrica. Inoltre, dato $x \in \mathbb{R}^n$ si ha che $\Pi x \in \mathcal{R}(Q)$, mentre se $x \perp \mathcal{R}(Q)$ allora $\Pi x = 0$; in particolare, per $x \in \mathbb{R}^n$ vale che $x = \Pi x + (I - \Pi)x := x_1 + x_2$, con $x_1 \perp x_2$. Infine, se le colonne di $Q$ sono solo linearmente indipendenti, la proiezione $\Pi$ assume la forma $\Pi = Q(Q^TQ)^{-1}Q^T$.

**Definizione 1.7.** *Sia $A \in \mathbb{K}^{m\times n}$. Si definisce la* ***norma di Frobenius*** *di $A$ come la quantità*

$$\|A\|_F := \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\right)^{\frac{1}{2}}.$$

*Osservazione* 1.8. La norma di Frobenius gode delle seguenti proprietà:

- $\|A^*\|_F = \|A\|_F$;
- $\|A\|_F = trace(A^*A)^{\frac{1}{2}}$;
- per ogni $U, V$ unitarie, $\|UAV^*\|_F = \|A\|_F$.

**Definizione 1.9.** *Sia $A \in \mathbb{K}^{m\times n}$. Si definisce la* ***norma-2*** *di $A$, indotta dalla norma vettoriale euclidea, come la quantità*

$$\|A\|_2 := \max_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1}\|Ax\|_2.$$

**Definizione 1.10.** *Sia $v \in \mathbb{K}^m$. Si definisce la* ***norma-0*** *di $v$ come il numero di elementi non nulli nel vettore.*

**Definizione 1.11.** *I sottospazi non banali $\{S_i\}_{i=1,\dots,M} \subset \mathbb{K}^m$ sono detti* ***indipendenti*** *se le loro dimensioni soddisfano la condizione*

$$dim(S_1 + \cdots + S_M) = dim(S_1) + \cdots + dim(S_M) \leq m.$$

Questa definizione è equivalente alla proprietà che ogni insieme di vettori non zero $\{w_1, \dots, w_M\}$ tali che $w_i \in S_i$, $i = 1, \dots, M$, sono linearmente indipendenti.

**Definizione 1.12.** *Sia $S$ un sottospazio vettoriale di $\mathbb{K}^m$ di dimensione $d$. Un insieme di dati $W$ tratti da $S$ si dice* ***generico*** *se*

- $|W| > d$;
- *qualsiasi insieme di $d$ vettori di $W$ forma una base per $S$.*

**Definizione 1.13.** *Un **frame** in uno spazio di Hilbert $\mathcal{V}$ è un insieme di vettori $\{v_i\}_{i\in\mathcal{I}} \subseteq \mathcal{V}$ con limiti del frame $0 < A \leq B < \infty$ tale che*

$$A\|x\|^2 \leq \sum_{i\in\mathcal{I}} |\langle x, v_i\rangle|^2 \leq B\|x\|^2 \qquad \forall x \in \mathcal{V}.$$

**Definizione 1.14.** *La **spark** di una matrice $W$ è la dimensione del più piccolo sottoinsieme linearmente dipendente di colonne, cioè*

$$\text{Spark}(W) = \min\{\|x\|_0 : Wx = 0,\ x \neq 0\}.$$

*Nel caso in cui le colonne di $W \in \mathbb{K}^{m\times n}$ siano tutte linearmente indipendenti si pone* $\text{Spark}(W) = m + 1$.

*Osservazione* 1.15. La Definizione 1.12 è equivalente ad affermare che le colonne di $W$ formano un frame per $S$ con spark $d + 1$.

**Definizione 1.16.** *Sia $W = [w_1, \ldots, w_n] \subset \mathbb{K}^m$ con colonne tratte dall'unione di sottospazi $\mathcal{U} = \bigcup_{i=1}^{M} S_i$. $\Xi_W$ è una **matrice di similarità** per $W$ se e solo se valgono le seguenti condizioni:*

- *$\Xi_W$ è simmetrica;*
- *$\Xi_W(i, j) \neq 0$ se e solo se $w_i$ e $w_j$ appartengono allo stesso sottospazio.*

**Definizione 1.17.** *Sia $A \in \mathbb{K}^{m\times n}$. La **versione valore assoluto** di $A$ è la matrice denotata con $abs(A) \in \mathbb{R}^{m\times n}$ che ha come elementi $abs(A)(i, j) = |A(i, j)|$.*

**Definizione 1.18.** *Sia $A \in \mathbb{K}^{m\times n}$. La **versione binaria** di $A$ è la matrice denotata con $bin(A) \in \mathbb{R}^{m\times n}$ che ha come elementi $bin(A)(i, j) = 1$ se $A(i, j) \neq 0$ e $bin(A)(i, j) = 0$ se $A(i, j) = 0$.*

## 1.2 Decomposizione in Valori Singolari

Una qualsiasi matrice può essere ridotta in forma diagonale tramite pre e post-moltiplicazione per matrici unitarie. Precisamente, vale il seguente risultato.

**Teorema 1.19.** *Siano $A \in \mathbb{C}^{m\times n}$ e $q = min\{m, n\}$. Allora esistono una matrice $\Sigma = diag(\sigma_1, \ldots, \sigma_q, 0, \ldots, 0) \in \mathbb{R}^{m\times n}$ con $\sigma_1 \geq \cdots \geq \sigma_q \geq 0$, e due matrici $U \in \mathbb{C}^{m\times m}, V \in \mathbb{C}^{n\times n}$ tali che*

$$A = U\Sigma V^*. \tag{1.1}$$

*La (1.1) è detta decomposizione in valori singolari (o Singular Value Decomposition, da cui l'abbreviazione SVD). I valori $\sigma_i$ sono chiamati valori singolari di $A$, mentre le colonne $u_i$ e $v_i$ di $U$ e $V$ sono rispettivamente note come vettori singolari sinistri e destri.*

*Dimostrazione.* Si rimanda a [6] per il caso reale. □

Nel caso in cui $A$ sia reale, allora anche $U$ e $V$ saranno reali e ortogonali e nell'equazione (1.1), $U^*$ andrà sostituita con $U^T$. In generale, si ha la seguente caratterizzazione per i valori singolari di $A$:

$$\sigma_i(A) = \sqrt{\lambda_i(A^*A)},\ i = 1, \dots, q.$$

In effetti dalla (1.1) si hanno $A = U\Sigma V^*$ e $A^* = V\Sigma^* U^*$, e quindi, essendo $U$ e $V$ unitarie, $A^*A = V\Sigma^*\Sigma V^*$ ovvero $\lambda_i(A^*A) = \lambda_i(\Sigma^*\Sigma) = (\sigma_i(A))^2$. Dal momento che $AA^*$ e $A^*A$ sono matrici hermitiane, le colonne di $U$ (rispettivamente di $V$) sono gli autovettori di $AA^*$ (rispettivamente di $A^*A$) e, di conseguenza, non sono definiti in modo univoco. Inoltre, la norma di Frobenius ammette una rappresentazione usando i valori singolari di $A$:

$$\|A\|_F := \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\right)^{\frac{1}{2}} = \left(\sum_i \sigma_i(A)^2\right)^{\frac{1}{2}}.$$

*Osservazione* 1.20. Per quanto riguarda il rango, se risulta

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_q = 0,$$

allora il rango di $A$ è $r$, il nucleo di $A$ è lo spazio generato dai vettori colonna di $V$, $\{v_{r+1}, \dots, v_n\}$, ed il range di $A$ è lo spazio generato dai vettori colonna di $U$, $\{u_1, \dots, u_r\}$.

In particolare, se $A = U\Sigma V^*$ è la SVD di $A$ e per $k < q$ vale $\sigma_{k+1} \ll \sigma_k$, allora:

$$A = \sum_{i=1}^{q} u_i\sigma_i v_i^* = \sum_{i=1}^{k} u_i\sigma_i v_i^* + \sum_{i=k+1}^{q} u_i\sigma_i v_i^* \approx \sum_{i=1}^{k} u_i\sigma_i v_i^* =: A_k. \tag{1.2}$$

La SVD è particolarmente utile perchè fornisce la migliore approssimazione a rango basso per una matrice, come spiegato nel seguente teorema.

**Teorema 1.21.** *Siano $A \in \mathbb{R}^{m\times n}$ e $A_k$ come in* (1.2)*. Allora valgono*

$$A_k = \underset{B\in\mathbb{R}^{m\times n},\ rank(B)=k}{\operatorname{argmin}} \|A - B\|_2, \qquad e \qquad \|A - A_k\|_2 = \sigma_{k+1}.$$

*Dimostrazione.* Sia $B$ una matrice di rango $k$ e sia $\{x_1, \dots, x_{n-k}\}$ una base per lo spazio nullo di $B$. Allora deve essere $Z = \{x_1, \dots, x_{n-k}\} \cap \{v_1, \dots, v_{k+1}\} \neq \{0\}$. Sia $z \in Z$ con $\|z\| = 1$. Dunque $Bz = 0$ e $Az = \sum_{i=1}^{k+1} u_i\sigma_i v_i^* z$. Quindi:

$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|\sum_{i=1}^{k+1} u_i\sigma_i v_i^* z\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^* z)^2 \geq \sigma_{k+1}^2.$$

D'altra parte, si ha che $k$ è il rango di $A_k$, ed essa raggiunge il minimo, infatti:

$$\|A - A_k\|_2 = \|\sum_{i>k} u_i \sigma_i v_i^*\|_2 = \sigma_{k+1}.$$

□

Nel seguito, la SVD troncata di ordine $k$ di $A$ sarà rappresentata da $A_k = U_k \Sigma_k V_k^*$, dove $U_k, V_k$ comprendono i primi $k$ vettori singolari rispettivamente sinistri e destri, e $\Sigma_k$ è una matrice $k \times k$ contenente i $k$ valori singolari più grandi. Inoltre, assumeremo sempre che i valori singolari siano disposti in ordine decrescente, ovvero $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q \geq 0$. Per specificare la matrice coinvolta, useremo la notazione $\sigma_i(A)$ per indicare l'i-esimo valore singolare di $A$.

## 1.3 Pseudoinversa di Moore-Penrose

**Definizione 1.22.** *Sia $A \in \mathbb{K}^{n\times n}$. $A$ si dice **invertibile** se esiste una matrice $B \in \mathbb{K}^{n\times n}$ tale che $AB = BA = I$. In tal caso $B$ viene chiamata **matrice inversa** di $A$ ed è indicata con $A^{-1}$.*

*Osservazione* 1.23. Se $A$ e $B$ sono due matrici invertibili di ordine $n$ allora anche $AB$ è invertibile e si ha $(AB)^{-1} = B^{-1}A^{-1}$.

La definizione di invertibiltà può essere estesa tramite la nozione di pseudo-inversa di Moore-Penrose, che ne rappresenta una generalizzazione. Esistono due definizioni della pseudo-inversa, la prima attribuita a Moore (1935) e la seconda a Penrose (1955).

**Definizione 1.24.** *Sia $A \in \mathbb{K}^{m\times n}$. L'**inversa generalizzata** di $A$ è l'unica matrice $A^\dagger \in \mathbb{K}^{n\times m}$ tale che*

1. $AA^\dagger = P_{\mathcal{R}(A)}$,
2. $A^\dagger A = P_{\mathcal{R}(A^\dagger)}$,

*ovvero $AA^\dagger$ è il proiettore ortogonale di $\mathbb{K}^m$ su $\mathcal{R}(A)$ e $A^\dagger A$ è il proiettore ortogonale di $\mathbb{K}^n$ su $\mathcal{R}(A^\dagger) = \mathcal{R}(A^*)$.*

**Definizione 1.25.** *Sia $A \in \mathbb{K}^{m\times n}$. L'**inversa generalizzata** di $A$ è l'unica matrice $A^\dagger \in \mathbb{K}^{n\times m}$ che soddisfa le seguenti condizioni:*

1. $AA^\dagger A = A$,
2. $A^\dagger AA^\dagger = A^\dagger$,

*3.* $(AA^\dagger)^* = AA^\dagger$,

*4.* $(A^\dagger A)^* = A^\dagger A$.

**Teorema 1.26.** *Le due definizioni di Moore e Penrose di inversa generalizzata sono equivalenti.*

*Dimostrazione.* Si veda [8]. □

Per ovvie ragioni, l'inversa generalizzata definita sopra viene spesso chiamata **pseudo-inversa di Moore-Penrose**. Si osserva che le equazioni (1) e (4) costituiscono effettivamente una sorta di legge di cancellazione: se normalmente $AB = AC$ non implica che $B = C$, possiamo affermare che se $A^\dagger AB = A^\dagger AC$ allora $AB = AC$.

Nel caso in cui $A \in \mathbb{K}^{n\times n}$ sia invertibile, la pseudo-inversa di Moore-Penrose coincide con la normale inversa $A^\dagger = A^{-1}$. Tuttavia, ci sono alcune proprietà che l'inversa generalizzata non possiede: ad esempio, se $A \in \mathbb{K}^{m\times n}$ e $B \in \mathbb{K}^{n\times p}$, allora $(AB)^\dagger$ non necessariamente coincide con $B^\dagger A^\dagger$. Inoltre, sappiamo che se $A$ e $C$ sono due matrici simili, allora condividono gli stessi autovalori, lo stesso polinomio caratteristico e la stessa forma di Jordan. D'altra parte, nessuna di queste proprietà è mantenuta considerando l'inversa generalizzata: l'unico fatto noto su $A^\dagger$ e $C^\dagger$ è che hanno lo stesso rango. Nonostante queste limitazioni, ci sono numerosi risultati utili basati sulla pseudo-inversa; introduciamo ora quelli di base, le cui dimostrazioni possono essere trovate in [8, capitolo 1].

**Teorema 1.27.** *Sia* $A \in \mathbb{K}^{m\times n}$. *Valgono le seguenti:*

*(I)* $(A^\dagger)^\dagger = A$

*(II)* $(A^\dagger)^* = (A^*)^\dagger$

*(III) dato* $\lambda \in \mathbb{K}$, *si ha* $(\lambda A)^\dagger = \lambda^\dagger A^\dagger$, *dove* $\lambda^\dagger = \frac{1}{\lambda}$ *se* $\lambda \neq 0$ *e* $\lambda^\dagger = 0$ *se* $\lambda = 0$

*(IV)* $A^* = A^*AA^\dagger = A^\dagger AA^*$

*(V)* $(A^*A)^\dagger = A^\dagger A^{*\dagger}$

*(VI)* $A^\dagger = (A^*A)^\dagger A^* = A^*(AA^*)^\dagger$

*(VII)* $(UAV)^\dagger = V^*A^\dagger U^*$, *dove* $U$ *e* $V$ *sono matrici unitarie.*

**Proposizione 1.28.** *Se $A$ è una matrice diagonale a blocchi,*

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & = & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & A_M \end{bmatrix}, \text{ allora } A^\dagger = \begin{bmatrix} A_1^\dagger & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & A_M^\dagger \end{bmatrix}.$$

**Teorema 1.29.** *Sia $A \in \mathbb{K}^{m\times n}$. Valgono le seguenti:*

*(I)* $\mathcal{R}(A) = \mathcal{R}(AA^\dagger) = \mathcal{R}(AA^*)$

*(II)* $\mathcal{R}(A^\dagger) = \mathcal{R}(A^*) = \mathcal{R}(A^\dagger A) = \mathcal{R}(A^*A)$

*(III)* $\mathcal{R}(I - AA^\dagger) = \mathcal{N}(AA^\dagger) = \mathcal{N}(A^*) = \mathcal{N}(A^\dagger) = \mathcal{R}(A)^\perp$

*(IV)* $\mathcal{R}(I - A^\dagger A) = \mathcal{N}(A^\dagger A) = \mathcal{N}(A) = \mathcal{R}(A^*)^\perp$.

**Proposizione 1.30.** *Sia $A \in \mathbb{K}^{m\times n}$. Allora*

- *se $n \geq m$ e $A$ ha rango $m$, allora la pseudo-inversa è la matrice $A^\dagger = (A^*A)^{-1}A^*$ ed è un'inversa sinistra, cioè $A^\dagger A = I$;*
- *se $n \leq m$ e $A$ ha rango $n$, allora la pseudo-inversa è la matrice $A^\dagger = A^*(AA^*)^{-1}$ ed è un'inversa destra, cioè $AA^\dagger = I$.*

Un modo efficiente per calcolare la pseudo-inversa $A^\dagger$ è attraverso la decomposizione in valori singolari di $A$. Infatti, se $A = U\Sigma V^*$ è la SVD della matrice $A$ di rango $r$, allora $A^\dagger = V\Sigma^\dagger U^*$, dove la pseudo-inversa di $\Sigma = diag(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbb{K}^{m\times n}$ è $\Sigma^\dagger = diag(1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0) \in \mathbb{K}^{n\times m}$.

**Definizione 1.31.** *Siano $A \in \mathbb{K}^{m\times n}$ e $b \in \mathbb{K}^m$. Un vettore $u \in \mathbb{K}^n$ è detto* ***soluzione ai minimi quadrati*** *del problema $Ax = b$ se $\|Au - b\| \leq \|Av - b\|$ per ogni $v \in \mathbb{K}^n$. Inoltre, $u \in \mathbb{K}^n$ è detto* ***soluzione ai minimi quadrati di norma minima*** *se è una soluzione ai minimi quadrati per $Ax = b$ e vale $\|u\| < \|w\|$ per ogni altra soluzione ai minimi quadrati $w$.*

Uno dei risultati più significativi riguardanti la pseudo-inversa è il fatto che essa consenta di determinare la soluzione ai minimi quadrati di norma minima per il problema $Ax = b$, come illustrato nel teorema seguente.

**Teorema 1.32.** *Siano $A \in \mathbb{K}^{m\times n}$ e $b \in \mathbb{K}^m$. Allora la soluzione ai minimi quadrati di norma-2 minima di $Ax = b$ è $A^\dagger b$.*

*Dimostrazione.* Sia $r$ il rango di $A$ e si consideri la decomposizione SVD di $A = U\Sigma V^*$. Per ogni $x \in \mathbb{K}^n$ si ha

$$\begin{aligned} \|Ax - b\|_2^2 &= \|U\Sigma V^* x - UU^* b\|_2^2 \\ &= \|\Sigma V^* x - U^* b\|_2^2 \\ &= \|\Sigma y - U^* b\|_2^2 \\ &= \sum_{i=1}^{r} (\sigma_i y_i - u_i^* b)^2 + \sum_{i=r+1}^{m} (u_i^* b)^2, \end{aligned} \tag{1.3}$$

dove $y = V^* x$. Si osserva che solo il primo termine dipende da $x$, quindi $\|Ax - b\|_2$ sarà minima scegliendo $y_i = (u_i^* b)/\sigma_i = (\Sigma U^* b)_i$ per $i = 1, \dots, r$. Questo fissa le prime $r$ componenti di $x_{LS}$ affinchè sia soluzione del problema, ora sarà sufficiente prendere le restanti componenti nulle per avere la soluzione di norma minima, perciò si avrà

$$x_{LS} = Vy = \sum_{i=1}^{r} \frac{u_i^* b}{\sigma_i} v_i = V\Sigma^\dagger U^* b = A^\dagger b.$$

Inoltre,

$$\|Ax_{LS} - b\|_2^2 = \sum_{i=r+1}^{m} (u_i^* b)^2.$$

□

## 1.4 Richiami sui Grafi

Nel seguente paragrafo, saranno introdotti alcuni concetti fondamentali della teoria dei grafi.

**Definizione 1.33.** *Sia $G = (V, E)$ un grafo con vertici nell'insieme $V = \{v_1, \dots, v_k\}$ e lati in $E$. Un **laccio** $l$ di $G$ è un lato i cui estremi coincidono, cioè tale che $l = (v_i, v_i)$ con $v_i \in V$.*

**Definizione 1.34.** *Sia $G = (V, E)$ un grafo con vertici nell'insieme $V = \{v_1, \dots, v_k\}$ e lati in $E$. Un **cammino** in $G$ è una sequenza alternata $(v_1, e_{12}, v_2, e_{23}, v_3, \dots, v_m)$ di vertici $\{v_1, v_2, v_3, \dots, v_m\}$ e lati $\{e_{12}, e_{23}, \dots\}$ che inizia e finisce con dei vertici, e tale che gli estremi di ogni lato siano i vertici che lo precedono e lo seguono. In un cammino i vertici possono ripetersi, invece se tutti i vertici sono distinti si parla di **percorso**.*

**Definizione 1.35.** *Sia $G = (V, E)$ un grafo con vertici nell'insieme $V = \{v_1, \dots, v_k\}$ e lati in $E$. La **distanza geodesica** tra due vertici $v_i, v_j \in V$ è la lunghezza (calcolata*

*come numero di vertici) del percorso più breve che connette $v_i$ e $v_j$. Il* ***diametro*** *del grafo $G$ è il massimo delle distanze geodesiche tra tutte le coppie di vertici in $G$, cioè $d(G) = \max_{v_i, v_j \in V} dist(v_i, v_j)$.*

**Definizione 1.36.** *Sia $G = (V, E)$ un grafo non diretto (ovvero simmetrico, non orientato) con vertici nell'insieme $V = \{v_1, \ldots, v_k\}$ e lati in $E$. $G$ si dice* ***pesato*** *se ad ogni lato $e_{ij}$ tra due vertici $v_i$ e $v_j$ è associato un peso $w_{ij}$. Inoltre, il grafo $G$ si dice* ***positivamente pesato*** *se $w_{ij} \geq 0$ per ogni $i, j$.*

**Definizione 1.37.** *Sia $G = (V, E)$ un grafo con vertici nell'insieme $V = \{v_1, \ldots, v_k\}$ e lati in $E$. $G$ è* ***connesso*** *se per ogni coppia di vertici $v_i$ e $v_j$ esiste un cammino che li congiunge. In particolare, $G$ si dice* ***fortemente connesso*** *se per ogni coppia di vertici $v_i$ e $v_j$ esiste un cammino diretto che connette $v_i$ e $v_j$.*

**Definizione 1.38.** *Sia $G = (V, E)$ un grafo con vertici nell'insieme $V = \{v_1, \ldots, v_k\}$ e lati in $E$. Ad ogni grafo è associata una* ***matrice di adiacenza*** *$A = (a_{ij})_{i,j=1,\ldots,k}$ tale che $a_{ij} = 1$ se c'è un lato tra $v_i$ e $v_j$, e $a_{ij} = 0$ altrimenti. In particolare, per un grafo non diretto, $A$ è simmetrica. Diciamo inoltre che $A$ è una* ***matrice di adiacenza positivamente pesata*** *di $G$ se $a_{ij} > 0$ se e solo se $v_i$ e $v_j$ sono connessi.*

**Teorema 1.39.** *Sia $G$ un grafo non diretto con vertici nell'insieme $V = \{v_1, \ldots, v_k\}$ e sia $A$ la matrice di adiacenza ad esso associata. Allora $A^r(i, j)$ conta il numero di cammini di lunghezza $r$ che connettono $v_i$ e $v_j$.*

*Dimostrazione.* Procediamo per induzione su $r$.

**Passo base**. Per definizione di matrice di adiacenza, se $r = 1$, si ha che $a_{ij} = 1$ se e solo se c'è un percorso di lunghezza uno tra $v_i$ e $v_j$.

**Passo induttivo**. Si consideri la matrice $A^{r+1} = A^r A$, e supponiamo che esista un cammino di lunghezza $r + 1$ tra due vertici arbitrari $v_m, v_l \in V$. Allora, esiste un $k_0$ tale che ci sia un cammino di lunghezza uno tra $v_{k_0}$ e $v_l$, e un cammino di lunghezza $r$ tra $v_m$ e $v_{k_0}$. Dunque $A(k_0, l) = 1$ per costruzione e $A^r(m, k_0) = r$ per ipotesi induttiva. Dato che $A^{r+1} = A^r A$, vale

$$A^{r+1}(m, l) = \sum_k A^r(m, k) A(k, l),$$

dove il $k$-esimo termine conta il numero di cammini di lunghezza $r+1$ tra $v_m$ e $v_l$ passanti per $v_k$. □

# Capitolo 2

# Decomposizione CUR e Subspace Clustering

## 2.1 Decomposizione CUR

### Decomposizione esatta

Data una matrice $A \in \mathbb{K}^{m \times n}$ di rango $r$, i metodi di fattorizzazione di matrici cercano di scrivere il prodotto $A = BX$, dove $B$ e $X$ sono matrici di struttura meno complessa. Nel caso in cui $B$ sia composta da colonne di $A$, si parla di *decomposizione interpolativa*, di cui un esempio è proprio la decomposizione CUR, in quanto un primo approccio ad essa si basa sulla scomposizione della matrice in termini di alcune delle sue colonne e righe. In particolare, scegliendo almeno $r$ colonne e righe di $A$ dovrebbe essere possibile ricostruire $A$ stessa, infatti il teorema seguente prova che per un'opportuna matrice $U$ è possibile ottenere una decomposizione CUR per $A$ che ne sia una fattorizzazione esatta, cioè tale che $A = CU^{\dagger}R$.

**Teorema 2.1.** *Siano $A \in \mathbb{K}^{m \times n}$ una matrice di rango $r$, $I \subseteq \{1, ..., m\}$ e $J \subseteq \{1, ..., n\}$ con $|I|=s$ e $|J|=k$; siano $C \in \mathbb{K}^{m \times k}$ avente per colonne le colonne di $A$ con indici in $J$, $R \in \mathbb{K}^{s \times n}$ avente per righe le righe di $A$ con indici in $I$, $U \in \mathbb{K}^{s \times k}$ la sottomatrice di $A$ con elementi con indici in $I \times J$. Se anche $U$ ha rango $r$, allora $A = CU^{\dagger}R$.*

*Dimostrazione.* Poichè $rank(U) = r$, si ha che $rank(R) = rank(C) = rank(A) = r$, quindi le colonne di $C$ formano una base per lo spazio delle colonne di $A$, e dunque $A = CX$ per qualche $X \in \mathbb{K}^{k \times n}$ (non necessariamente unica). Sia $P_I \in \mathbb{K}^{s \times m}$ una matrice che seleziona le righe con indici in $I$, cioè tale che $R = P_I A$. Allora $R = P_I A = P_I CX$; si nota inoltre che $U = P_I C$, quindi si ha $R = UX$. Poichè anche $rank(R) = r$, ogni soluzione di $R = UX$ è anche soluzione di $A = CX$. Allora la soluzione del teorema segue

osservando che $Y = U^+R$ è soluzione di $R = UX$. In effetti, un ragionamento analogo al precedente implica che $Y = U^+R$ è soluzione di $R = UX$ se e solo se è soluzione di $RP_J = U = UXP_J$, dove $P_J \in \mathbb{K}^{n\times k}$ è una matrice che seleziona le colonne con indici in $J$, cioè tale che $C = AP_J$. Dunque, osservando che $Y = U^+R$ è soluzione di $U = UXP_J$, infatti $U = UU^+RP_J = UU^+U = U$, si ha la tesi, ovvero $A = CU^\dagger R$. □

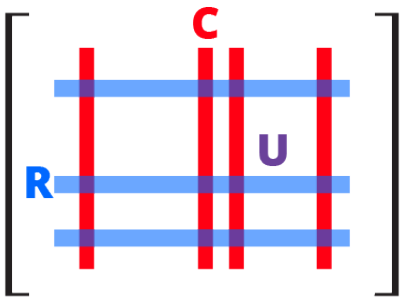


**Figura 2.1:** *Illustrazione della decomposizione CUR dal punto di vista delle matrici. $C$ è la sottomatrice rossa con le colonne di $A$, mentre $R$ è la sottomatrice blu di $A$, e $U$ è la loro intersezione in viola.*

*Osservazione* 2.2. Si osserva che l'ipotesi $rank(U) = r$ nel teorema precedente implica che $k, s \geq r$.

Il Teorema 2.1 mostra la decomposizione CUR nella forma più generale. Tuttavia, in alcuni casi, essa si riduce a una versione più semplice, come si evince dai seguenti corollari, ciascuno dei quali segue dalla particolare forma della pseudoinversa $U^\dagger$ sotto determinate ipotesi di indipendenza sulle righe e sulle colonne.

**Corollario 2.3.** *Siano $A, C, U, R$ come nel Teorema 2.1; in particolare, siano le colonne di $C \in \mathbb{K}^{m\times r}$ linearmente indipendenti. Allora $A = C(U^*U)^{-1}U^*R$.*

**Corollario 2.4.** *Siano $A, C, U, R$ come nel Teorema 2.1; in particolare, siano le righe di $R \in \mathbb{K}^{r\times n}$ linearmente indipendenti. Allora $A = CU^*(UU^*)^{-1}R$.*

**Corollario 2.5.** *Siano $A, C, U, R$ come nel Teorema 2.1; in particolare, sia $U \in \mathbb{K}^{r\times r}$ invertibile. Allora $A = CU^{-1}R$.*

## Approssimazione di rango basso

L'obiettivo è ora di individuare un'approssimazione di rango basso di $A$, ovvero una matrice di rango piccolo (rispetto alle dimensioni della matrice stessa) che sia idealmente vicina ad $A$ in norma. Come già dimostrato nel Teorema 1.21, fissato $k < rank(A)$, la migliore approssimazione di rango $k$ di $A$ è la decomposizione SVD troncata, ma essa spesso non conserva alcune caratteristiche della struttura della matrice iniziale $A$. Ad esempio, se $A$ è sparsa, le matrici $U, V$, che contengono rispettivamente i vettori singolari sinistri e destri, saranno dense, così come le loro versioni troncate. Inoltre, un aspetto cruciale nell'analisi di dati è l'interpretabilità dei risultati, e nella SVD troncata i vettori singolari $u_i$ e $v_i$ sono quantità "astratte" matematiche che non sempre hanno un significato utile nell'ambito da cui i dati sono tratti.

Dai motivi appena elencati nasce l'idea di selezionare solo poche colonne rappresentative della matrice di dati $A$ che contengano tutte le informazioni necessarie su $A$; in altre parole, si desidera proiettare $A$ nello spazio generato da un sottoinsieme ridotto di colonne rappresentative in modo che il risultato sia vicino ad $A$ in norma. Ricordando quello che è già stato dimostrato e il fatto che $CC^\dagger$ è il proiettore nello spazio delle colonne di $C$, questo problema, noto come *Column Subset Selection Problem* può essere formulato come segue: dati una matrice $A \in \mathbb{K}^{m\times n}$ e un $k \leq n$ fissato, si vuole trovare una sottomatrice formata da colonne $C$ che risolva

$$\min_{\substack{C=A(:,J)\\ J\subset\{1,\dots,n\},|J|=k}} \|A - CC^\dagger A\|_F.$$

Lo stesso principio può essere applicato alle righe della matrice, ovvero scegliere quelle che contengono maggiori informazioni sullo spazio delle righe di $A$, e questo problema può essere riassunto così: dati $A \in \mathbb{K}^{m\times n}$ e $k \leq m$, si vuole trovare una sottomatrice formata da righe $R$ che risolva

$$\min_{\substack{C=A(I,:)\\ I\subset\{1,\dots,m\},|I|=k}} \|A - AR^\dagger R\|_F.$$

Si osserva che risolvere il problema sulle righe equivale a risolvere il *Column Subset Selection Problem* sulle colonne di $A^*$. A questo punto è naturale unire questi due problemi e dunque cercare di trovare una matrice $Z$ che minimizzi la quantità $\|A - CZR\|_F$.

**Proposizione 2.6.** *Siano $A \in \mathbb{K}^{m\times n}$ e $C, R$ sottomatrici formate rispettivamente da colonne e righe di $A$. Allora vale*

$$\operatorname*{argmin}_{Z\in\mathbb{K}^{k\times s}} \|A - CZR\|_F = C^\dagger A R^\dagger.$$

*Dimostrazione.* Per il Teorema 1.32, è noto che la soluzione di norma minima di

$$\min_{X\in\mathbb{K}^{k\times n}} \|A - CX\|_F$$

è $\tilde{X} = C^\dagger A$. Riapplicando il Teorema 1.32 al problema

$$\min_{U\in\mathbb{K}^{k\times s}} \|\tilde{X} - UR\|_F = \min_{U\in\mathbb{K}^{k\times s}} \|\tilde{X}^* - R^*U^*\|_F,$$

si ha che la sua soluzione di norma minima è data da

$$\tilde{U^*} = (R^*)^\dagger \tilde{X}^* = (R^\dagger)^* \tilde{X}^* = (\tilde{X}R^\dagger)^*$$

da cui $\tilde{U} = \tilde{X}R^\dagger = C^\dagger AR^\dagger$. □

Basandosi sul risultato ottenuto nella proposizione precedente, spesso la decomposizione CUR viene definita come $A \approx CC^\dagger AR^\dagger R$.

## Confronto tra decomposizione esatta e approssimazione

Lo scopo è ora confrontare i due approcci visti alla decomposizione CUR. Il risultato principale, dimostrato nel Teorema 2.11, è che nel caso della decomposizione esatta questi punti di vista sono effettivamente lo stesso e portano alla medesima conclusione. Inoltre, vengono fornite diverse caratterizzazioni equivalenti per determinare quando si ottiene una decomposizione CUR esatta. Prima di enunciare questo teorema, facciamo nota di alcune considerazioni utili sulle matrici coinvolte.

**Lemma 2.7.** *Siano $A, C, U, R$ come nel Teorema 2.1, con $rank(A) = rank(U)$. Allora $\mathcal{N}(C) = \mathcal{N}(U)$, $\mathcal{N}(R^*) = \mathcal{N}(U^*)$, $\mathcal{N}(A) = \mathcal{N}(R)$, e $\mathcal{N}(A^*) = \mathcal{N}(C^*)$. Inoltre,*

$$C^\dagger C = U^\dagger U, \qquad RR^\dagger = UU^\dagger,$$

$$AA^\dagger = CC^\dagger, \qquad A^\dagger A = R^\dagger R.$$

*Dimostrazione.* Come nella dimostrazione del Teorema 2.1, la condizione $rank(U) = rank(A) = k$ implica che anche $C$ e $R$ hanno rango $k$, e da ciò seguono direttamente tutti i risultati relativi allo spazio nullo delle matrici $A, C, U, R$.

Adesso, si ricorda che se $C \in \mathbb{K}^{m\times s}$, allora l'applicazione $C^\dagger C : \mathbb{K}^s \to \mathbb{K}^s$ è la proiezione ortogonale su $\mathcal{N}(C)^\perp$. D'altra parte, $\mathcal{N}(C) = \mathcal{N}(U)$ e perciò $\mathcal{N}(C)^\perp = \mathcal{N}(U)^\perp$. Di conseguenza, $C^\dagger C$ è la proiezione ortogonale su $\mathcal{N}(U)^\perp$, ma questa è $U^\dagger U$. Allo stesso modo, $RR^\dagger$ è la proiezione ortogonale su $\mathcal{N}(R^*)^\perp = \mathcal{N}(U^*)^\perp$, da cui $RR^\dagger = UU^\dagger$, e si ottiene la tesi. Le ultime due uguaglianze seguono applicando lo stesso ragionamento. □

**Lemma 2.8.** *Siano $A \in \mathbb{K}^{m\times n}$ e $B \in \mathbb{K}^{n\times p}$. Se $rank(A) = rank(B) = n$, allora $rank(AB) = n$.*

*Dimostrazione.* Segue dalla disuguaglianza di Sylvester

$$rank(A) + rank(B) - n \leq rank(AB).$$

□

Nelle dimostrazioni dei prossimi due risultati la decomposizione SVD di $A$ sarà indicata come $A = W\Sigma V^*$, con l'uso di $W$ al posto della tipica $U$, poichè quest'ultima è utilizzata come matrice centrale nella decomposizione CUR.

**Corollario 2.9.** *Sia $A \in \mathbb{K}^{m\times n}$ con $rank(A) = k$ e siano $C = A(:, J) \in \mathbb{K}^{m\times k}$ e $R = A(I, :) \in \mathbb{K}^{s\times n}$ con $rank(C) = rank(R) = k$. Allora posto $U = A(I, J) \in \mathbb{K}^{s\times k}$ si ha $rank(U) = k$.*

*Dimostrazione.* Se la SVD troncata di rango $k$ di $A$ è $A = W_k\Sigma_k V_k^*$, allora $C = W_k\Sigma_k(V_k(J, :))^*$. Dato che $rank(C) = k$, si ha $rank(V_k(J, :)) = k$ e, con lo stesso ragionamento, si ottiene anche $rank(W_k(I, :)) = k$. Infine $U = A(I, J) = W_k(I, :)\Sigma_k(V_k(J, :))^*$, dunque per il Lemma 2.8, si ha che $rank(U) = k$. □

**Proposizione 2.10.** *Siano $A, C, U, R$ come nel Teorema 2.1 (ma senza alcuna ipotesi sul rango di $U$). Allora*

$$U = RA^\dagger C.$$

*Dimostrazione.* Consideriamo la decomposizione in valori singolari di $A$, $A = W\Sigma V^*$. Allora si hanno $C = A(:, J) = W\Sigma V^*(:, J)$, $R = A(I, :) = W(I, :)\Sigma V^*$, e $U = W(I, :)\Sigma V^*(:, J)$. Di conseguenza, si ottiene

$$\begin{aligned} RA^\dagger C &= W(I, :)\Sigma V^* V\Sigma^\dagger W^* W\Sigma V^*(:, J) \\ &= W(I, :)\Sigma\Sigma^\dagger\Sigma V^*(:, J) \\ &= W(I, :)\Sigma V^*(:, J) = U. \end{aligned}$$

□

La proposizione precedente propone una forma per $U$ che differisce dall'intersezione tra $C$ e $R$. Abbiamo ora tutti gli strumenti per enunciare il teorema fondamentale.

**Teorema 2.11.** *Siano $A \in \mathbb{K}^{m\times n}$, $I \subseteq \{1, ..., m\}$ e $J \subseteq \{1, ..., n\}$ (anche con elementi ripetuti). Siano $C = A(I, :)$, $U = A(I, J)$, e $R = A(:, J)$. Allora le seguenti sono equivalenti:*

*(I)* $rank(U) = rank(A)$

*(II)* $A = CU^{\dagger}R$

*(III)* $A = CC^{\dagger}AR^{\dagger}R$

*(IV)* $A^{\dagger} = R^{\dagger}UC^{\dagger}$

*(V)* $rank(C) = rank(R) = rank(A)$.

*Dimostrazione.* Si procede come segue

(I)⇒(II) Segue dal Teorema 2.1.

(II)⇒(I) Per assurdo, si suppone che $rank(U) \neq rank(A)$, ma per come viene costruita $U$, ciò implica che $rank(U) < rank(A)$. D'altra parte,

$$rank(A) = rank(CU^{\dagger}R) \leq rank(U^{\dagger}) = rank(U) < rank(A),$$

che è una contraddizione. Perciò $rank(U) = rank(A)$.

(I)⇒(V) Immediata.

(V)⇒(I) Segue dal Corollario 2.9.

(V)⇒(III) Applicando il Lemma 2.7 si hanno $AA^{\dagger} = CC^{\dagger}$ e $A^{\dagger}A = R^{\dagger}R$, e si giunge facilmente alla tesi grazie alle proprietà della pseudoinversa di Moore-Penrose:

$$A = AA^{\dagger}A = AA^{\dagger}AA^{\dagger}A = CC^{\dagger}AR^{\dagger}R.$$

(III)⇒(V) Per costruzione $Span(C) \subset Span(A)$, ma (III) implica che $Span(A) \subset Span(C)$. Con un ragionamento analogo si ottiene che $Span(R^{*}) = Span(A^{*})$, da cui $(V)$.

(I)⇒(IV) Applicando la Proposizione 2.10, il Lemma 2.7 e le proprietà della pseudoinversa si ottiene la tesi:

$$R^{\dagger}UC^{\dagger} = R^{\dagger}RA^{\dagger}CC^{\dagger} = A^{\dagger}AA^{\dagger}AA^{\dagger} = A^{\dagger}.$$

(IV)⇒(I) Per la Proposizione 2.10, si ha

$$A = AA^{\dagger}A = AR^{\dagger}UC^{\dagger}A = AR^{\dagger}RA^{\dagger}CC^{\dagger}A.$$

Dunque $rank(A) = rank(AR^{\dagger}RA^{\dagger}CC^{\dagger}A) \leq rank(C) \leq rank(A)$, da cui $rank(A) = rank(C)$. Similmente si prova che $rank(A) = rank(R)$. Infine, utilizzando il Corollario 2.9 si conclude che $rank(U) = rank(A)$.

□

Questo teorema fornisce diverse caratterizzazioni per la decomposizione CUR esatta e, in particolare, l'equivalenza tra le condizioni (II) e (III) dimostra che i due approcci proposti coincidono nel caso della decomposizione esatta (e in nessun altro caso). Successivamente, vengono proposte ulteriori osservazioni e conseguenze derivanti da questo risultato.

**Proposizione 2.12.** *Siano $A, C, U, R$ come nel Teorema 2.1, con $rank(A) = rank(U)$. Allora*

$$U^\dagger = C^\dagger A R^\dagger.$$

*Dimostrazione.* Per il Teorema 2.1 si ha $A = CU^\dagger R$. Dunque per il Lemma 2.7 e le proprietà della pseudoinversa,

$$C^\dagger A R^\dagger = C^\dagger C U^\dagger R R^\dagger = U^\dagger U U^\dagger U U^\dagger = U^\dagger.$$

□

*Osservazione* 2.13. La condizione nella Proposizione 2.12 non è sufficiente; se $U^\dagger = C^\dagger A R^\dagger$, allora $rank(A)$ non sempre coincide con $rank(U)$. Ad esempio, sia

$$A = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix},$$

e siano $C = \begin{bmatrix} 0 \\ I \end{bmatrix}$ e $R = \begin{bmatrix} 0 & I \end{bmatrix}$. Allora $U = U^\dagger = 0$, e $C^\dagger A R^\dagger = \begin{bmatrix} 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ I \end{bmatrix} = 0$. Perciò $U^\dagger = C^\dagger A R^\dagger$, ma $rank(A) \neq rank(U)$.

*Osservazione* 2.14. La Proposizione 2.10 vale per qualsiasi scelta di sottomatrici $C$ e $R$ rispettivamente di colonne e righe, mentre affinchè valga la Proposizione 2.12 è necessaria l'ulteriore ipotesi che $U$ e $A$ abbiano lo stesso rango. Senza questo presupposto sul rango, la Proposizione 2.12 non segue immediatamente della Proposizione 2.10, e ciò è dato dal fatto che in generale $(AB)^\dagger$ non coincide con $B^\dagger A^\dagger$. Inoltre, la conclusione della Proposizione 2.12 non garantisce che il rango di $U$ sia uguale a quello di $A$, e di conseguenza non implica nessuna delle condizioni elencate nel Teorema 2.11.

Un altro vantaggio della decomposizione CUR esatta è che offre una forma utile per la pseudoinversa di Moore-Penrose, esplicitata nella seguente proposizione.

**Proposizione 2.15.** *Siano $A, C, U, R$ come nel Teorema 2.1, con $rank(A) = rank(U)$. Allora*

$$A^\dagger = R^\dagger U C^\dagger.$$

*Dimostrazione.* Vedi (I)⇒(IV) nella dimostrazione del Teorema 2.11. □

Per concludere questa sezione, mostriamo che nel caso dell'approssimazione CUR (in particolare quando viene scelto un numero di colonne e righe di $A$ minore del suo rango), le matrici $C^\dagger AR^\dagger$ e $U^\dagger$ sono diverse, e inoltre la scelta di $C^\dagger AR^\dagger$ come matrice centrale crea una migliore approssimazione rispetto a $U^\dagger$.

*Esempio* 2.16. Si considera la matrice a rango pieno

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},$$

e siano

$$C = A(:,1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \text{ e } R = A(1,:) = \begin{bmatrix} 1 & 2 \end{bmatrix}.$$

Allora $U = U^\dagger = 1$, e si osserva che $\|A - CU^\dagger R\|_F = 2$. D'altra parte, cercando il minimo della funzione $f(z) = \|A - CzR\|_F$ si ottiene un valore minimo di 1.0583 con $z = 0.76$, che corrisponde a $C^\dagger AR^\dagger$.

## 2.2 Subspace Clustering

In questa sezione, viene introdotto il problema del Subspace Clustering e viene esaminato come la decomposizione CUR possa essere utilizzata per affrontarlo.

**Ipotesi 1.** *Annotiamo alcune ipotesi generali a cui faremo riferimento in seguito. Supponiamo che* $\mathcal{U} = \bigcup_{i=1}^{M} S_i$ *sia un insieme non lineare che consiste nell'unione di sottospazi* $\{S_i\}_{i=1,\dots,M}$ *di* $\mathbb{K}^m$ *non banali, lineari e indipendenti, con corrispondenti dimensioni* $\{d_i\}_{i=1,\dots,M}$*, e poniamo* $d_{max} := \max_{1 \le i \le M} d_i$*. Inoltre, assumiamo che la matrice di dati* $W = [w_1, \dots, w_n] \in \mathbb{K}^{m \times n}$ *abbia colonne tratte da* $\mathcal{U}$*, e che i dati provenienti da ogni sottospazio* $S_i$ *siano generici per quel sottospazio.*

### Subspace Clustering

La necessità di elaborare sempre più grandi quantità di dati ha spinto lo sviluppo e l'avanzamento del clustering tradizionale, portando all'introduzione di nuove tecniche più sofisticate, come il *Subspace Clustering*, che può infatti esserne considerato un'estensione. Molte tecniche di clustering si basano sull'osservazione che, anche se i dataset hanno una dimensione elevata, la loro dimensione intrinseca è spesso molto più piccola della dimensione dello spazio ambiente. Le tecniche convenzionali, come l'Analisi delle Componenti Principali (PCA), assumono che i dati siano estratti da un unico sottospazio

a bassa dimensione di uno spazio ad alta dimensione. Approcci di questo tipo hanno trovato ampie applicazioni in molti campi, ma nella pratica i dati potrebbero essere tratti da più sottospazi e potrebbe non essere noto a quale sottospazio i dati appartengano. Ad esempio, in un video potrebbero esserci più oggetti in movimento e questo renderebbe necessario l'utilizzo di diversi sottospazi per descrivere i diversi moti. Pertanto, c'è l'esigenza di raggruppare simultaneamente i dati in più sottospazi e trovare un sottospazio a bassa dimensione per ciascun gruppo di punti.

Questo problema, noto come *Subspace Clustering*, ha trovato numerose applicazioni nella visione artificiale (segmentazione del moto, riconoscimento facciale) e nell'elaborazione delle immagini (rappresentazione e compressione delle immagini). In particolare, la formulazione del problema del *Subspace Clustering* può essere sintetizzata in questo modo: se $W = [w_1, \dots, w_n] \in \mathbb{K}^{m \times n}$ è una matrice di dati che soddisfa le Ipotesi 1, quindi tratta dall'unione di sottospazi, l'obiettivo è determinare il numero di sottospazi, la loro dimensione e una base per ciascuno di essi, al fine di individuare i diversi clusters. In questo elaborato viene esaminato un caso semplificato del *Subspace Clustering* aggiungendo l'ipotesi che i sottospazi siano indipendenti.

## L'uso della decomposizione CUR nel subspace clustering

Uno dei principali approcci al *Subspace Clustering* consiste nel generare una matrice di similarità a partire dai dati, dalla quale sia possibile leggere immediatamente i clusters, almeno quando i dati soddisfano le Ipotesi 1 e non sono affetti da rumore. Il Teorema 2.23 mostra che, a partire dalla CUR della matrice dei dati $W$, è possibile ottenere molteplici matrici di similarità, e questo perchè la decomposizione CUR non è unica (le righe e colonne selezionate possono variare). Prima di presentare il teorema menzionato, esponiamo alcuni risultati preliminari necessari per la sua dimostrazione.

**Lemma 2.17.** *Sia $W \in \mathbb{K}^{m \times n}$ una matrice di dati generici tratti da un sottospazio $S$ di dimensione $r < n$. Si consideri la fattorizzazione $W = BP$, dove le colonne di $B \in \mathbb{K}^{m \times r}$ formano una base per il range di $W$. Sia $P = [p_1, \dots, p_n] \in \mathbb{K}^{r \times n}$, dove per ogni $i = 1, \dots, n$, $p_i$ rappresentano le colonne di $P$. Allora, le colonne di $P$ sono generiche per $\mathbb{K}^r$, ovvero ogni insieme di $r$ colonne di $P$ forma una base per $\mathbb{K}^r$.*

*Dimostrazione.* Siano $p_{i_1}, \dots, p_{i_r}$ $r$ colonne di $P$ e supponiamo che esistano delle costanti $c_1, \dots, c_r \in \mathbb{K}$ tali che $c_1 p_{i_1} + c_2 p_{i_2} + \cdots + c_r p_{i_r} = 0$. Allora $B(c_1 p_{i_1} + c_2 p_{i_2} + \cdots + c_r p_{i_r}) = 0$, da cui $c_1 w_{i_1} + c_2 w_{i_2} + \cdots + c_r w_{i_r} = 0$. Dato che $W$ è generica per $S$, deve essere $c_i = 0$ per ogni $i = 1, \dots, r$, dunque le colonne $p_{i_1}, \dots, p_{i_r}$ sono linearmente indipendenti. □

**Lemma 2.18.** *Sia $U \in \mathbb{K}^{m\times n}$ con colonne generiche per il sottospazio $S \subset \mathbb{K}^m$ dal quale sono tratte. Sia $P \in \mathbb{K}^{r\times m}$ una matrice che seleziona le righe tale che $rank(PU) = rank(U)$. Allora le colonne di $PU$ sono generiche per $S$.*

*Dimostrazione.* É un caso particolare del Lemma 2.17, infatti si osserva che vale la scrittura $U = BPU$, dove le colonne di $B$ formano una base per $\mathbb{K}^m$ per l'ipotesi $rank(PU) = rank(U)$. □

**Lemma 2.19.** *Sia $U = [U_1, \dots, U_M]$, dove ogni $U_i$ è una sottomatrice le cui colonne sono tratte da sottospazi indipendenti $S_i$, $i = 1, \dots, M$ di $\mathbb{K}^m$, e sono generiche per $S_i$. Sia $P \in \mathbb{K}^{r\times m}$ con $r \geq rank(U)$ una matrice che seleziona le righe tale che $rank(PU) = rank(U)$. Allora i sottospazi $P(S_i)$, $i = 1, \dots, M$ sono indipendenti.*

*Dimostrazione.* Sia $S = S_1 + \cdots + S_M$, e siano $d_i = \dim(S_i)$ e $d = \dim(S)$. Dalle ipotesi, otteniamo che $rank(PU_i) = d_i$, e che $rank(PU) = d = \sum_{i=1}^{M} d_i$. Dunque, ci sono $d$ vettori linearmente indipendenti per $P(S)$ nelle colonne di $PU$; in particolare, dato che $PU = [PU_1, \dots, PU_M]$, ci sono $d_i$ vettori linearmente indipendenti per $P(S_i)$ nelle colonne di $PU_i$. Perciò, $\dim P(S) = d = \sum_i d_i = \sum_i \dim P(S_i)$, e si ha la tesi. □

L'ingrediente chiave nella dimostrazione del Teorema 2.23 risiede nel fatto che la matrice $Y = U^\dagger R$, che genera la matrice di similarità, è diagonale a blocchi grazie alla struttura composta da sottospazi indipendenti di $\mathcal{U}$. Questa caratteristica viene illustrata nel Teorema 2.20, che segue immediatamente.

**Teorema 2.20.** *Sia $W = [W_1, \dots, W_n] \in \mathbb{K}^{m\times n}$ una matrice di rango $r$ le cui colonne sono tratte da $\mathcal{U}$ che soddisfa le Ipotesi 1. Sia $W = CU^+R$ fattorizzata come nel Teorema 2.1, con $C \in \mathbb{K}^{m\times k}$, $R \in \mathbb{K}^{s\times n}$, $U \in \mathbb{K}^{s\times k}$. Se $Y = U^\dagger R$, allora esiste una matrice di permutazione $P$ tale che*

$$YP = \begin{bmatrix} Y_1 & 0 & \cdots & 0 \\ 0 & Y_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & Y_M \end{bmatrix},$$

*dove per ogni $i = 1, \dots, M$ $Y_i \in \mathbb{K}^{k_i\times n_i}$, con $n_i$ il numero di colonne in $W$ tratte dal sottospazio $S_i$, e $k_i$ il numero di colonne in $C$ tratte da $S_i$. Inoltre $WP = [W_1, \dots, W_M]$, dove le colonne di $W_i$ sono tratte dal sottospazio $S_i$.*

*Dimostrazione.* Senza perdita di generalità, possiamo assumere che $W = [W_1, \dots, W_M]$, dove per ogni $i = 1, \dots, M$ $W_i \in \mathbb{K}^{m\times n_i}$, con $\sum_{i=1}^{M} n_i = n$, e che i vettori colonna di $W_i$ siano tratti dal sottospazio $S_i$. Mostriamo che $Y$ è una matrice diagonale a blocchi. Sia

$P$ la matrice che seleziona le righe, cioè tale che $PW = R$; in particolare, si osserva che, poichè $P : \mathbb{R}^m \to \mathbb{R}^s$ e per la struttura di $W$, è possibile scrivere $R = [R_1, \dots, R_M]$ dove le colonne di $R_i$ appartengono al sottospazio $\tilde{S}_i := P(S_i)$. Si osserva anche che le colonne di $R_i$ sono generiche per il sottospazio $\tilde{S}_i$ per il Lemma 2.18, e che i sottospazi $\tilde{S}_i$ sono indipendenti per il Lemma 2.19. Inoltre, dato che $U$ è composta da certe colonne di $R$ e $rank(U) = rank(R) = rank(W)$ per ipotesi, si ha che $U = [U_1, \dots, U_M]$, dove le colonne di $U_i$ appartengono a $\tilde{S}_i$. Ora, dalla dimostrazione del Teorema 2.1 sulla decomposizione CUR, sappiamo che $Y = U^\dagger R$ è soluzione di $R = UX$, cioè si ha $R = UY$. Sia $r$ una colonna di $R_1$, e sia $y = [y_1, y_2, \dots, y_M]^*$ la corrispondente colonna di $Y$ tale che $r = Uy$. Dunque $r = \sum_{j=1}^{M} U_j y_j$, ed essendo una colonna di $R_1$, si ottiene

$$(U_1 y_1 - r) + U_2 y_2 + \cdots + U_M y_M = 0,$$

e a questo punto l'indipendenza dei sottospazi $\tilde{S}_i$ implica che $U_1 y_1 = r$ e $U_i y_i = 0$ per ogni $i = 2, \dots, M$. D'altra parte, anche $\tilde{y} = [y_1, 0, \dots, 0]^*$ è un'altra soluzione, cioè vale $r = U\tilde{y}$; ricordando che $U^\dagger r$ è la soluzione di minima norma del problema $r = Uy$, si deve avere

$$\|y\|_2^2 = \sum_{i=1}^{M} \|y_i\|_2^2 \leq \|\tilde{y}\|_2^2 = \|y_1\|_2^2.$$

Di conseguenza, $y = \tilde{y}$, e si conclude che $Y$ sia diagonale a blocchi applicando lo stesso argomento alle colonne di $R_i$, $i = 2, \dots, M$. □

**Lemma 2.21.** *Sia $G$ un grafo finito, connesso, non diretto e positivamente pesato con diametro $r$ in cui ognuno dei vertici $\{v_1, \dots, v_N\}$ ha un laccio, e sia $A$ la matrice d'adiacenza associata. Allora $A^r(i, j) > 0$ per ogni $i, j$. In particolare, $A^r$ è la matrice di adiacenza di un grafo fortemente connesso.*

*Dimostrazione.* Ricordando che per il Teorema 1.39 $A^r(i, j)$ conta il numero di cammini di lunghezza $r$ tra $v_i$ e $v_j$, mostriamo per induzione che se $A^r(i, j) > 0$ allora c'è un cammino di lunghezza $r$ tra $v_i$ e $v_j$.

**Passo base**. Per costruzione, $A(i, j) > 0$ se e solo se c'è un cammino di lunghezza uno tra $v_i$ e $v_j$.

**Passo induttivo**. Si considerino due vertici $v_m$ e $v_l$. Allora, come nella dimostrazione del Teorema 1.39 si ha

$$A^{r+1}(m, l) = \sum_k A^r(m, k) A(k, l) > 0.$$

Dunque esiste un $k$ tale che $A^r(m, k) > 0$ e $A(k, l) > 0$. Di conseguenza, ci sono un percorso di lunghezza $r$ tra $v_m$ e $v_l$ e un percorso di lunghezza uno tra $v_k$ e $v_l$. Infine,

essendo $r$ il diametro di $G$, tra ogni coppia di vertici si può trovare sempre un percorso di lunghezza al più $r$ che li unisca sfruttando la connessione del grafo e il fatto che ogni vertice abbia un laccio. □

**Lemma 2.22.** *Sia $V = \{p_1, \dots, p_N\}$ un insieme di vettori generici che rappresentano dati tratti da un sottospazio $S$ di dimensione $r$, con $N > r \geq 1$. Sia $Q$ come nel Teorema 2.23 nel caso $\mathcal{U} = S$. Infine, sia $G$ il grafo con vertici $p_i$ e lati $p_ip_j$ se $Q(i,j) > 0$. Allora $G$ è connesso, e ha diametro al più $r$. Inoltre, $Q^r$ è la matrice di adiacenza di un grafo fortemente connesso.*

*Dimostrazione.* Per dimostrare che $G$ è connesso, distinguiamo due casi.

**N=r+1**. Sia $C$ un insieme non vuoto di vertici di una componente connessa di $G$. Per assurdo, si supponga che $C^C$ contenga $1 \leq k \leq r$ vertici. Dunque, dato che $N = r + 1$, si ha $|C^C| \leq r$, e quindi i vettori $\{p_j\}_{j \in C^C}$ sono linearmente indipendenti per l'ipotesi di genericità. Allo stesso modo, $|C| \leq r$, perciò anche i vettori $\{p_i\}_{i \in C}$ sono linearmente indipendenti. D'altra parte, per costruzione di $G$, $\langle p, q \rangle = 0$ per ogni $p \in C$ e $q \in C^C$, dunque l'insieme $\{p_i\}_{i \in C \cup C^C}$ è un insieme di $r + 1$ vettori in $S$ linearmente indipendenti, assurdo perchè la dimensione di $S$ è $r$. Di conseguenza, $C^C$ è vuoto e $G$ è connesso.

**N>r+1**. Siano $p \neq q$ elementi qualsiasi di $V$, mostriamo che esiste un percorso che connette $p$ e $q$. Poichè $N > r$ e $V$ è generico, esiste un insieme $V_0 \subset V \setminus \{p, q\}$ di cardinalità $r - 1$ tale che $\{p, q\} \cup V_0$ è un insieme di vettori generici in $S$. Questo rappresenta un sottografo di $r + 1$ vertici di $G$ che soddisfa le condizioni del caso base $N = r + 1$, quindi questo sottografo è connesso. Perciò, c'è un percorso che connette $p$ e $q$, e per la loro arbitrarietà si può concludere che $G$ è connesso.

In particolare, nella dimostrazione del caso generale $N > r + 1$, è stato mostrato che esiste un percorso di lunghezza al più $r$ che connette due vertici arbitrari $p, q$, e quindi il diametro di $G$ è al più $r$. L'ultimo risultato segue direttamente dal Lemma 2.21. □

**Teorema 2.23.** *Sia $W = [w_1, \dots, w_n] \in \mathbb{K}^{m \times n}$ una matrice di rango $r$ le cui colonne sono tratte da $\mathcal{U}$ che soddisfa le Ipotesi 1. Sia $W = CU^\dagger R$ fattorizzata come nel Teorema 2.1, con $C \in \mathbb{K}^{m \times k}$, $R \in \mathbb{K}^{s \times n}$, e $U \in \mathbb{K}^{s \times k}$. Siano $Y = U^\dagger R$ e $Q$ la versione valore assoluto o la versione binaria di $Y^*Y$. Allora, $\Xi_W = Q^{d_{max}}$ è una matrice di similarità per $W$.*

*Dimostrazione.* Senza perdita di generalità, per il Teorema 2.20 possiamo assumere che $Y$ sia diagonale a blocchi, mostriamo che ogni blocco $Y_i$ ha rango $d_i = \dim(S_i)$ ed ha colonne generiche per $S_i$. Dato che $R_i = U_iY_i$ e $rank(R_i) = rank(U_i) = d_i$, si ha

$rank(Y_i) \geq d_i$ poichè il rango di un prodotto di matrici è al più il minimo dei ranghi. D'altra parte, poichè $Y_i = U^\dagger R_i$, si ha $rank(Y_i) \leq rank(R_i) = d_i$, quindi $rank(Y_i) = d_i$. Per provare che le colonne di ogni $Y_i$ sono generiche, siano $y_1, \ldots, y_{d_i}$ $d_i$ colonne in $Y_i$ e supponiamo che esistano delle costanti $c_1, \ldots, c_{d_i} \in \mathbb{K}$ tali che $\sum_{j=1}^{d_i} c_j y_j = 0$. Per via della struttura diagonale a blocchi di $Y$ ne consegue che

$$0 = U_i \left( \sum_{j=1}^{d_i} c_j y_j \right) = \sum_{j=1}^{d_i} c_j U_i y_j = \sum_{j=1}^{d_i} c_j r_j,$$

dove $r_j$, per $j = 1, \ldots, d_i$, sono le colonne di $R_i$. Essendo le colonne di $R_i$ generiche per il Lemma 2.18, si ha che $c_j = 0$ per ogni $j = 1, \ldots, d_i$, di conseguenza le colonne di $Y_i$ sono generiche.

A questo punto, $Q = Y^*Y \in \mathbb{K}^{n \times n}$ è una matrice diagonale a blocchi con blocchi $Q_i = Y_i^* Y_i$, $i = 1, \ldots, M$, dove ogni blocco $Q_i$ può essere considerato la matrice di adiacenza di un grafo che soddisfa le ipotesi del Lemma 2.21. Dunque, per quanto già dimostrato, ogni blocco rappresenta un grafo connesso con diametro $d_i$, e perciò $Q^{d_{\max}}$ dà origine a un grafo con $M$ componenti fortemente connesse distinte, dove il grafo derivante da $Q_i$ corrisponde ai dati in $W$ tratti da $S_i$. Quindi $Q^{d_{\max}}$ è una matrice di similarità per $W$. □

**Corollario 2.24.** *Sia $W$ come nel Teorema 2.23, fattorizzata in $W = CU^\dagger R$. Se $C = W$, allora $Q = |R^\dagger R|$, dove $Q$ è stata costruita come nel Teorema 2.23.*

**Corollario 2.25.** *Sia $W$ come nel Teorema 2.23, fattorizzata in $W = CU^\dagger R$. Se $C = R = W$, allora $Q = |W^\dagger W|$, dove $Q$ è stata costruita come nel Teorema 2.23.*

*Osservazione* 2.26. Nel teorema precedente, $Q$ è quasi una matrice di similarità, ma ogni cluster potrebbe non essere fortemente connesso; elevare la matrice a $d_{max}$ assicura quest'ultima condizione.

# Capitolo 3

# Applicazioni al Subspace Clustering

## 3.1 Algoritmo

Consideriamo il caso, molto comune nelle applicazioni, di dati con rumore, in cui la matrice dei dati $W$ è una piccola perturbazione di una matrice di dati ideale, cioè le colonne di $W$ sono approssimativamente tratte dall'unione di sottospazi, ovvero $w_i = u_i + \eta_i$, dove $u_i \in \mathcal{U}$ e $\eta_i$ è un piccolo vettore di rumore. In questo elaborato, ci limitiamo a considerare l'applicazione della teoria descritta in precedenza agli ambiti della segmentazione del moto e del riconoscimento facciale. Ad esempio, nella segmentazione del moto, la matrice di dati iniziale contiene nelle colonne le traiettorie di diversi oggetti in movimento, ma dimostreremo che le traiettorie delle caratteristiche relative allo stesso corpo rigido si trovano in un sottospazio di dimensione molto inferiore rispetto alla dimensione dello spazio. Vale un ragionamento analogo anche per il riconoscimento facciale, rendendo questi due problemi pratici adatti per verificare la validità dell'approccio proposto.

Alcuni punti dell'algoritmo possono risultare ambigui nell'implementazione, dunque cerchiamo di spiegare meglio:

- **Passo 2**. Si fattorizza $W$ con le decomposizione CUR come descritto nel Teorema 2.1. Ricordando che la decomposizione CUR non è unica, questo punto ammette varie possibilità di implementazione. Ci sono infatti diversi modi per scegliere colonne e righe nella decomposizione CUR, ad esempio in [5] vengono selezionate casualmente in accordo con una distribuzione di probabilità che si basa su dei "valori di importanza" associati a colonne e righe. Nel nostro caso, scegliamo colonne e righe casualmente con probabilità uniforme. Questo passaggio permette ancora maggiore flessibilità perchè può variare anche il numero di colonne e righe scelte.

**Algoritmo:** CUR Subspace Clustering

**Input:** Una matrice di dati $W = [w_1, \dots, w_N] \in \mathbb{K}^{m \times n}$, il numero atteso di sottospazi $M$, e il numero di iterazioni $k$

**Output:** Etichette dei clusters dei sottospazi

**for** *i = 1,...,k* **do**
  Fare la decomposizione CUR approssimata di $W$, $W \approx C_i U_i^\dagger R_i$
  $Y_i = U_i^\dagger R_i$
  Mettere delle soglie a $Y_i$
  Calcolare $\Xi_W^{(i)} = Y_i^* Y_i$
  Forzare connessioni note
**end**
$\Xi_W = \text{abs}(\text{median}(\Xi_W^{(1)}, \dots, \Xi_W^{(k)}))$
Fare clustering sulle colonne di $\Xi_W$
**return** etichette dei clusters

- **Passo 3**. Si calcola $Y_i$ in accordo con il Teorema 2.20, cioè come prodotto di $U_i^\dagger$ e $R_i$.

- **Passo 4**. Poichè per il Teorema 2.20 $Y_i$ è una matrice diagonale a blocchi, poniamo a zero gli elementi di $Y_i$ che in valore assoluto sono al di sotto di una certa soglia (0,001) ottenuta sperimentalmente, in modo che siano soddisfatte le condizioni di indipendenza dei sottospazi. In alternativa, se è noto a priori che gli $M$ sottospazi hanno tutti la stessa dimensione, allora $Y_i$ dovrebbe avere esattamente una proporzione di $1 - \frac{1}{M}$ di suoi elementi totali nulli. Pertanto, possiamo ordinare gli elementi in ordine decrescente, mantenere i primi $(1 - \frac{1}{M})\times$ (# totale di elementi) dall'alto e porre il resto uguale a zero.

- **Passo 5**. Si calcola $\Xi_W^{(i)}$ come il prodotto $Y_i^* Y_i$, senza applicare le potenze come suggerito dal Teorema 2.23. Questo è giustificato dal fatto che, in presenza di rumore nella matrice di similarità, prendere il prodotto tra matrici amplifica il rumore e peggiora significativamente la performance dell'algoritmo di clustering.

- **Passo 6**. Si utilizza tutta la conoscenza disponibile sui dati per migliorare il clustering finale. In generale, l'unica informazione nota a priori è il fatto che ovviamente la colonna $w_i$ si trova nello stesso sottospazio di $w_i$ stessa, perciò poniamo tutti gli elementi diagonali di $\Xi_W^{(i)}$ uguali a 1 per assicurarci che questa connessione sia garantita.

- **Passo 8**. Si calcola il valore assoluto della mediana delle matrici di similarità prodotte. La matrice di similarità derivata da una singola approssimazione CUR di una matrice di dati può contenere errori, per questo eseguiamo la media o mediana elemento per elemento della famiglia $\{\Xi_W^{(i)}\}_{i=1,\dots,k}$.

- **Passo 9**. Si applica un algoritmo di clustering a $\Xi_W$. A questo punto, $\Xi_W$ è una matrice di similarità non ideale, ovvero che non fornirà esattamente i clusters perchè alcuni suoi elementi sono nulli. Tuttavia, ogni colonna $j$ dovrebbe avere un elemento $(i, j)$ grande quando $w_i$ e $w_j$ appartengono allo stesso sottospazio, e piccola (in valore assoluto) altrimenti. Dunque un semplice algoritmo di clustering come le $k$-medie o lo spectral clustering dovrebbe restituire i clusters corretti. Nella prossima sezione, introduciamo inoltre un algoritmo di clustering che verrà utilizzato in questo passaggio e confrontato con gli algoritmi di $k$-medie e spectral clustering.

## 3.2 Principal Coordinate Clustering

Attualmente, esistono numerosi metodi per affrontare il problema del clustering in spazi di grandi dimensioni, e questi permettono di individuare gruppi significativi tra i dati, ovvero i clusters. Per implementare un algoritmo di clustering è spesso necessaria una matrice di similarità: ad esempio, nello spectral clustering si considera come matrice di similarità la matrice di adiacenza di un grafo non diretto e pesato. A partire da quest'ultima si costruisce poi la matrice Laplaciana, semi-definita positiva; successivamente, se ci sono $M$ gruppi, si applica un tradizionale algoritmo di clustering (come le $k$-medie) alla matrice che ha per colonne gli $M$ autovettori della Laplaciana associati agli $M$ autovalori più piccoli. Lo spectral clustering può risultare di difficile interpretazione, pertanto introduciamo un altro algoritmo di clustering particolarmente utile quando i dati sono tratti dall'unione di sottospazi indipendenti: il *Principal Coordinate Clustering* (PCC), efficace quanto lo spectral clustering, ma che risulta più intuitivo e di facile comprensione. Prende il nome dall'Analisi delle Componenti Principali, in quanto si basa sulla decomposizione a valori singolari della matrice di similarità.

*Osservazione* 3.1. Si osserva che, essendo la matrice di similarità $\Xi_W$ simmetrica, essa ha SVD ridotta di rango $M$ nella forma $\Xi_W = U_M \Sigma_M U_M^T$. Perciò il PCC esegue il clustering sulle colonne di $\Sigma_M U_M^T$, che rappresentano le coordinate principali rispetto alla base ridotta $u_1, \dots, u_M$ contenuta nelle colonne di $U_M$.

**Algoritmo 2:** Principal Coordinate Clustering

**Input:** Una matrice di similarità $\Xi_W$ per $W \in \mathbb{K}^{m \times n}$, e il numero di clusters $M$

**Output:** Etichette dei clusters

1 Calcolare la SVD ridotta di rango $M$ di $\Xi_W$, $\Xi_W = U_M \Sigma_M V_M^T$

2 Calcolare $\Sigma_M V_M^T \in \mathbb{K}^{M \times n}$

3 Applicare l'algoritmo delle $k$-medie, con $k = M$, alle colonne di $\Sigma_M V_M^T$

4 **return** etichette dei clusters

*Osservazione* 3.2. Nello spectral clustering, uno degli scopi per cui viene calcolata la matrice Laplaciana è ottenere una matrice semi-definita positiva, mentre il PCC non richiede alcun vincolo di questo tipo.

# Capitolo 4

# Applicazioni ed esperimenti numerici

In questo capitolo, esaminiamo le applicazioni pratiche della decomposizione CUR, concentrandoci su due problemi di grande rilevanza: la segmentazione del moto e il riconoscimento facciale. Entrambi questi problemi rappresentano sfide significative nell'ambito dell'analisi dei dati e del machine learning, dove la capacità di identificare e raggruppare correttamente elementi simili è fondamentale.

La segmentazione del moto è un problema chiave nel campo della visione artificiale e della robotica. Si tratta di identificare e separare oggetti in movimento all'interno di una scena, utilizzando dati derivati da video o sequenze di immagini. Per affrontare questo problema, utilizziamo il dataset Hopkins155.

Il riconoscimento facciale, invece, è una delle applicazioni più diffuse dell'intelligenza artificiale. In questo contesto, la decomposizione CUR è impiegata per migliorare l'efficienza e l'accuratezza del riconoscimento, riducendo la dimensionalità dei dati mantenendo al contempo le caratteristiche essenziali che distinguono i volti. Utilizziamo il Database of Faces per testare l'efficacia dell'approccio proposto.

Attraverso questi esempi, dimostriamo come la decomposizione CUR possa essere applicata con successo a problemi reali, offrendo un valido metodo per la riduzione dimensionale e il clustering dei dati.

## 4.1 Segmentazione del moto

La segmentazione del moto a partire dai dati sulle traiettorie rappresenta una questione essenziale nella comprensione e ricostruzione delle scene dinamiche. Una scena dinamica è composta da più oggetti in movimento, con una telecamera statica o in mo-

to, e l'obiettivo è segmentare e distinguere le traiettorie delle caratteristiche in base ai movimenti nella scena. In generale, nella segmentazione del moto si parte da un video, e un algoritmo iniziale individua le caratteristiche degli oggetti in movimento e traccia la posizione di queste caratteristiche nel tempo. Alla fine del video, avremo ottenuto una matrice di dati di dimensione $2F \times N$, dove:

- $F$ è il numero di frames nel video;
- $N$ è il numero di caratteristiche tracciate.

Ogni vettore corrisponde alla traiettoria di una certa caratteristica, quindi è nella forma $(x_1, y_1, \ldots, x_F, y_F)^T$, dove $(x_i, y_i)$ è la posizione della caratteristica al frame $i$-esimo, con $1 \leq i \leq F$. Anche se queste traiettorie si trovano in uno spazio ambiente di grande dimensione $\mathbb{R}^{2F}$, è noto che le traiettorie delle caratteristiche di un certo corpo appartengono ad un sottospazio di dimensione molto minore di $2F$. Di conseguenza, la segmentazione del movimento di una scena dinamica, composta da più movimenti, sia indipendenti, articolati, rigidi, non rigidi, degeneri o non degeneri, può essere considerata come un problema di ricerca di sottospazi lineari, in cui l'incognita è la dimensionalità di questi sottospazi.

In particolare, dimostriamo ora che le traiettorie di tutte le caratteristiche relative allo stesso corpo rigido si trovano in un sottospazio di dimensione 4, secondo l'approccio in [11]. Si supponga di aver tracciato $N$ punti caratteristici di oggetti in movimento nel corso di $F$ immagini (frames), e siano $(x_{k\alpha}, y_{k\alpha})$ le coordinate dell'$\alpha$-esimo punto al $k$-esimo frame. Consideriamo nuovamente il vettore che rappresenta la traiettoria dell'$\alpha$-esimo punto:

$$p_\alpha = (x_{1\alpha}, y_{1\alpha}, x_{2\alpha}, y_{2\alpha}, \ldots, x_{F\alpha}, y_{F\alpha})^T \in \mathbb{R}^{2F}.$$

Prendiamo il sistema di coordinate XYZ della videocamera come il sistema di riferimento globale dove l'asse Z viene preso lungo l'asse ottico della videocamera stessa. Adesso, fissiamo un arbitrario oggetto in movimento e posizioniamo un sistema di coordinate relativo all'oggetto nell'immagine $k$-esima, denotato come $(t_k, \{i_k, j_k, k_k\})$, dove $t_k$ è l'origine e $\{i_k, j_k, k_k\}$ è la base ortonormale. Siano $(a_\alpha, b_\alpha, c_\alpha)$ le coordinate dell'$\alpha$-esimo punto rispetto al sistema di riferimento dell'oggetto. La sua posizione al $k$-esimo frame rispetto al sistema di coordinate globale è data da

$$r_{k\alpha} = t_k + a_\alpha i_k + b_\alpha j_k + c_\alpha k_k.$$

Dunque si ha

$$\begin{pmatrix} x_{k\alpha} \\ y_{k\alpha} \end{pmatrix} = \tilde{t}_k + a_\alpha \tilde{i}_k + b_\alpha \tilde{j}_k + c_\alpha \tilde{k}_k,$$

dove $\tilde{t}_k, \tilde{i}_k, \tilde{j}_k$ e $\tilde{k}_k$ sono i vettori bidimensionali ottenuti rispettivamente da $t_k, i_k, j_k$ e $k_k$ troncando la terza componente. A questo punto impiliamo verticalmente i vettori $\tilde{t}_k, \tilde{i}_k, \tilde{j}_k$ e $\tilde{k}_k$ per $k = 1, \ldots, F$, ovvero lungo gli $F$ frames, per creare rispettivamente i vettori bidimensionali $m_0, m_1, m_2$ e $m_3$, cioè si hanno

$$\begin{aligned} m_0 &= (\tilde{t}_1, \ldots, \tilde{t}_F), \\ m_1 &= (\tilde{i}_1, \ldots, \tilde{i}_F), \\ m_2 &= (\tilde{j}_1, \ldots, \tilde{j}_F), \\ m_3 &= (\tilde{k}_1, \ldots, \tilde{k}_F). \end{aligned}$$

Perciò il vettore $p_\alpha$ sarà nella forma

$$p_\alpha = m_0 + a_\alpha m_1 + b_\alpha m_2 + c_\alpha m_3. \tag{4.1}$$

Di conseguenza, gli $N$ punti $\{p_\alpha\}$ appartengono al sottospazio generato dai vettori $\{m_0, m_1, m_2, m_3\}$, di dimensione 4. Quindi in generale, se ci sono $M$ moti indipendenti, i punti $\{p_\alpha\}$ si trovano in un sottospazio di $\mathbb{R}^{2F}$ di dimensione $4M$. Pertanto, stimare il numero di moti indipendenti si riduce al calcolo del rango di un insieme di vettori, ovvero alla determinazione della dimensione del sottospazio da essi generato. Si osserva inoltre che per distinguere $M$ moti indipendenti sono necessarie almeno $2M$ immagini. Infatti, se si ha un moto rigido bidimensionale sul piano dell'immagine, il vettore $m_3$ nell'equazione (4.1) risulta nullo. Dunque il rango di $\{p_\alpha\}$ è $3M$ per $M$ moti indipendenti, e sono necessarie $1,5M$ immagini o più per distinguerli.

## Dataset Hopkins155

Nell'ambito della segmentazione del moto, è possibile applicare l'algoritmo di clustering con la decomposizione CUR al dataset Hopkins155, che contiene 155 video appartenenti a diverse categorie (tra parentesi è indicato il numero di video per ogni categoria):

- Checkerboard sequences (104): si tratta di sequenze registrate in un ambiente controllato, dove agli oggetti ripresi è stato sovrapposto un pattern a scacchiera con l'obiettivo di tracciare il maggior numero possibile di caratteristiche. Le sequenze includono sia situazioni con tre oggetti in movimento (due oggetti e la telecamera) sia con due oggetti (dove la telecamera è fissa), che possono subire rotazioni, traslazioni, o combinazioni di entrambi;

- Traffic Sequences (38): questo gruppo consiste in scene di traffico realizzate all'aperto e con la telecamera in movimento, e comprende numerosi esempi di moti degeneri, come moti lineari o su un piano;
- Articulated/non-rigid sequences (13): contiene ad esempio scene con movimenti delle mani o della testa oppure delle persone che camminano, dove gli oggetti in moto sono vincolati tra loro, quindi in questo caso non è più valida l'assunzione che tutte le caratteristiche relative allo stesso corpo rigido siano contenute in un sottospazio di dimensione 4.

Ad ogni video è associata una matrice di dati contenente le traiettorie di tutte le caratteristiche degli oggetti in movimento. Di conseguenza, la matrice dei dati è unica per ogni video, e viene fornito anche un vettore contenente le etichette corrette del clustering, permettendo così di calcolare la percentuale di errore di classificazione, definita ovviamente come la percentuale di caratteristiche assegnate ai clusters errati. Alcuni esempi di frames tratti dai video nel dataset Hopkins155 sono mostrati nella Figura 4.1.

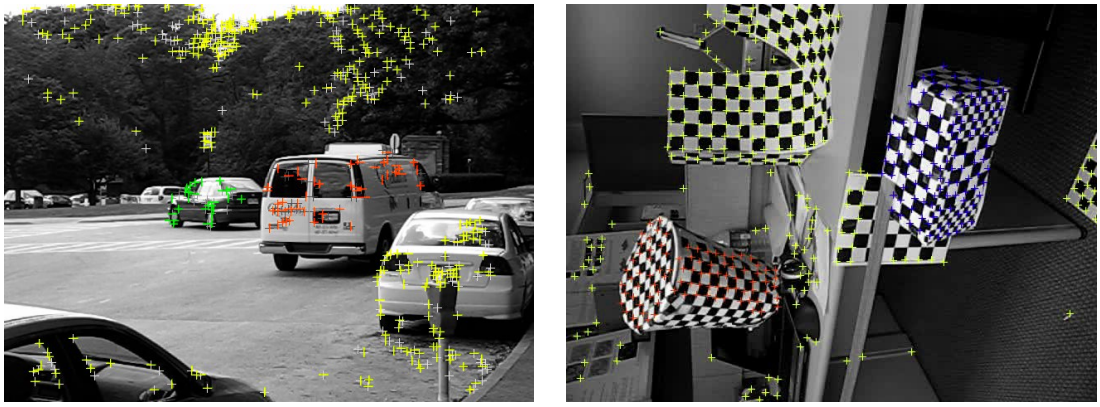

**Figura 4.1:** *Immagini tratte dal dataset Hopkins. Sinistra: traffico. Destra: scacchiera*

Nella decomposizione CUR (linea 2 dell'Algoritmo 1) scegliamo tutte le colonne e un numero di righe uguale al rango atteso di $R$, ovvero $4\times$(# di sottospazi $M$). Di conseguenza, si ottiene $Y = R^{\dagger}R$, dove le righe vengono selezionate da $W$ con una distribuzione di probabilità uniforme. Infatti si osserva sperimentalmente che questa decomposizione $W = WR^{\dagger}R$ mostra migliori risultati rispetto alla forma generale $W = CU^{\dagger}R$, e questo è giustificato dal fatto che ogni rappresentazione del tipo $W = CX$ è una rappresentazione di $W$ in termini dei suoi vettori colonna, e scegliere un maggior numero di colonne significa aggiungere ridondanza, che porta maggiore robustezza al rumore. Infine, in conformità con il Corollario 2.24, si avrà $Q = |R^{\dagger}R| \in \mathbb{K}^{n\times n}$, e sarà necessario assicurarsi che il numero di righe scelto, $4M$, sia strettamente inferiore a $n$, altrimenti,

come indicato nella Proposizione 1.30, la pseudoinversa di $R$ sarebbe un'inversa sinistra, e quindi $Q = I$, che non dà alcuna informazione utile per il clustering.

Nelle Tabelle 4.1-4.2 sono riportate le percentuali di errore di classificazione date dalla media di 10 runs dell'algoritmo sull'intero dataset (al fine di limitare la variabilità dei risultati che si ottengono con una singola esecuzione) effettuando $k = 50$ iterazioni: nella tabella 4.1 si considera $\Xi_W^{(i)} = Y_i^* Y_i$, mentre nella tabella 4.2 si ha $\Xi_W^{(i)} = (Y_i^* Y_i)^p$ con $p = 4$, ed effettivamente si osserva che l'elevamento a potenza peggiora leggermente le prestazioni dell'algoritmo. Notiamo inoltre che usare il Principal Coordinate Clustering su $\Xi_W$ ha prodotto risultati migliori rispetto agli algoritmi di spectral clustering e $k$-medie, così come la mediana è preferita alla media perchè assicura maggiore robustezza agli outliers.

| | **K-medie** | | **Spectral Clustering** | | **PCC** | |
|---|---|---|---|---|---|---|
| | **Media** | **Mediana** | **Media** | **Mediana** | **Media** | **Mediana** |
| **Checkerboard** | 6,05% | 5,95% | 2,07% | 2,63% | 1,66% | 1,59% |
| **Traffic** | 5,76% | 5,71% | 4,04% | 3,71% | 1,41% | 1,22% |
| **Articulated** | 7,50% | 7,19% | 4,69% | 5,08% | 6,64% | 6,32% |
| **Tutti** | 6,36% | 6,29% | 2,93% | 3,03% | 1,98% | 1,92% |

Tabella 4.1: % di errore per 10 runs con $k = 50$ e $\Xi_W^{(i)} = Y_i^* Y_i$.

| | **K-medie** | | **Spectral Clustering** | | **PCC** | |
|---|---|---|---|---|---|---|
| | **Media** | **Mediana** | **Media** | **Mediana** | **Media** | **Mediana** |
| **Checkerboard** | 6,11% | 6,25% | 2,24% | 2,85% | 1,68% | 1,66% |
| **Traffic** | 6,47% | 6,36% | 4,09% | 3,77% | 1,41% | 1,25% |
| **Articulated** | 8,11% | 7,19% | 4,84% | 5,23% | 6,73% | 6,54% |
| **Tutti** | 6,50% | 6,39% | 3,00% | 3,31% | 2,05% | 1,92% |

Tabella 4.2: % di errore per 10 runs con $k = 50$ e $\Xi_W^{(i)} = (Y_i^* Y_i)^p$, $p = 4$.

Nella Tabella 4.3, sono riportate le percentuali di errore e i tempi medi di 10 runs dell'algoritmo CUR confrontati con quelli degli algoritmi di spectral clustering e $k$-medie applicati direttamente ai dati iniziali $W$.

É inoltre possibile implementare un modo per determinare il numero di iterazioni $k$ in base alla matrice dei dati: per ogni $i$, calcoliamo $\tilde{\Xi}_W^{(i)} = \text{abs}(\text{median}(\Xi_W^{(1)}, \ldots, \Xi_W^{(i)}))$ la media delle matrici di similarità finora prodotte, poi calcoliamo $\|\tilde{\Xi}_W^{(i)} - \tilde{\Xi}_W^{(i-1)}\|_2$ e la confrontiamo con una soglia (come $0,01$). Se la norma è minore della soglia interrompiamo

| | Decomposizione CUR | K-medie | Spectral Clustering |
|---|---|---|---|
| % errore | 1,9177% | 14,8792% | 25,4574% |
| Tempo | 0,7376s | 0,0158s | 0,0212s |

Tabella 4.3: Confronto % di errore e tempo per 10 runs.

il ciclo, altrimenti continuiamo fino a un numero massimo di iterazioni $k_{max}$ prefissato. Per il dataset Hopkins155 vengono eseguite in media 60 iterazioni, da un minimo di 39 a un massimo di 100 (la soglia $k_{max}$). Osservando la Figura 4.2, che mostra la percentuale di errore e il tempo medi dell'algoritmo CUR in funzione di $k$, notiamo che dopo circa 50 iterazioni la performance dell'algoritmo peggiora, mentre il tempo di esecuzione aumenta significativamente. Perciò in questo caso si effettuano $k = 50$–$60$ iterazioni, ovvero decomposizioni CUR, per avere un clustering accurato in un tempo ragionevole.

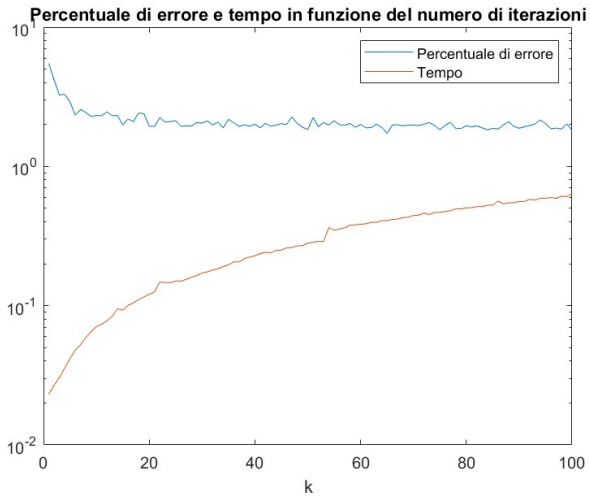


**Figura 4.2:** *Confronto percentuale di errore e tempo all'aumentare di k*

## 4.2 Riconoscimento facciale

Il riconoscimento facciale è un'importante applicazione della biometria basata sull'individuazione di un pattern che consente di classificare un volto mediante il confronto con altre immagini di persone presenti nel dataset. Tuttavia, esso rappresenta una sfida difficile da affrontare a causa della grande variabilità delle informazioni data dalle differenze tra individui, incluse variazioni casuali e sistematiche come quelle legate all'illuminazione e alla posa. In particolare, l'identificazione della posa è considerata un

aspetto complesso perchè tutti i volti sono simili, nel senso che ogni faccia consiste in due occhi, un naso, una bocca e altre caratteristiche che generalmente occupano la stessa posizione. Nonostante questo, il volto umano è una struttura estremamente complessa e dinamica, con caratteristiche che possono cambiare in modo significativo e rapido nel tempo. Inoltre, mentre gli esseri umani hanno una capacità innata di riconoscere i volti, il loro numero può rendere difficile la memorizzazione; pertanto, un vantaggio chiave di un sistema automatico per il riconoscimento facciale è la sua capacità di memorizzare e gestire grandi quantità di dati, nonché di ridurre la dimensione dello spazio studiato.

Ogni immagine può essere considerata una matrice la cui dimensione è data dal numero di pixels. Questa matrice viene quindi riorganizzata disponendo le sue colonne, una dopo l'altra, in un vettore colonna: ad esempio, nel Database of Faces le immagini sono di dimensione 92x112 (pixels), quindi ognuna di esse è rappresentata da un vettore in $\mathbb{R}^{10304}$. Di conseguenza, la matrice di dati associata a un dataset di foto viene costruita mettendo le varie immagini come colonne, e perciò essa avrà dimensione $n_1 n_2 \times N$, dove:

- $n_1 n_2$ è la dimensione del vettore colonna che rappresenta l'immagine, ottenuto dal prodotto del numero di pixels;
- $N$ è il numero totale di immagini nel dataset.

Anche per il riconoscimento facciale, le immagini relative ad una stessa persona si trovano in un sottospazio di dimensione molto inferiore rispetto alla dimensione totale delle immagini $n_1 n_2$, il che rende anche questo problema idoneo all'approccio proposto.

## Dataset Database of Faces

Nel contesto del riconoscimento facciale, è possibile applicare l'algoritmo di clustering mostrato al dataset Database of Faces, precedentemente noto come ORL Database. Esso contiene un serie di immagini di volti scattate tra aprile 1992 e aprile 1994 in un laboratorio. In particolare, sono disponibili 10 diverse immagini per ciascuno dei 40 soggetti presenti nel dataset; per alcuni soggetti le immagini sono state acquisite in momenti diversi e variando le condizioni di illuminazione, l'espressione facciale (come occhi aperti o chiusi e sorridenti o meno) e altri dettagli (come portare o meno gli occhiali). Infine, tutte le fotografie sono state scattate con uno sfondo scuro e uniforme, con il soggetto in posizione eretta e frontale (consentendo una lieve tolleranza per i movimenti laterali). Alcuni esempi di immagini presenti nel dataset sono mostrate nella Figura 4.3.

Nell'Algoritmo 1 utilizziamo gli stessi parametri della segmentazione del moto; in particolare, nella decomposizione CUR selezioniamo tutte le colonne e un numero di righe uguali a $3\times$(# di sottospazi $M$), che è risultata empiricamente la scelta più accurata.

**Figura 4.3:** *Immagini tratte dal dataset Database of Faces*

Nelle Tabelle 4.4-4.5 sono riportate le percentuali di errore di classificazione e i tempi dati dalla media di 10 runs dell'algoritmo con la mediana, il Principal Coordinate Clustering e $k = 150$ iterazioni. Questi risultati sono confrontati con quelli ottenuti con gli algoritmi delle $k$-medie e dello spectral clustering applicati ai dati iniziali. Nella Tabella 4.4 sono considerati i primi 10 soggetti del dataset, mentre nella Tabella 4.5 sono considerati i primi 20 soggetti. Si osserva che, anche in questo caso, la percentuale di errore generata dall'algoritmo con la decomposizione CUR mostra migliori risultati rispetto agli algoritmi di spectral clustering e $k$-medie, sebbene richieda più tempo di esecuzione.

| | **Decomposizione CUR** | **K-medie** | **Spectral Clustering** |
|---|---|---|---|
| **% errore** | 1,7000% | 13,5000% | 25,2000% |
| **Tempo** | 0,5162s | 0,1133s | 0,0118s |

Tabella 4.4: Confronto % errore e tempo per 10 runs per i primi 10 soggetti.

| | **Decomposizione CUR** | **K-medie** | **Spectral Clustering** |
|---|---|---|---|
| **% errore** | 9,6000% | 20,3500% | 41,5000% |
| **Tempo** | 2,3132s | 0,4519s | 0,0194s |

Tabella 4.5: Confronto % errore e tempo per 10 runs per i primi 20 soggetti.

Inoltre, utilizzando l'implementazione proposta precedentemente per il numero di iterazioni $k$, con la soglia $k_{max} = 200$, notiamo che per il dataset Database of Faces vengono eseguite in media 132 iterazioni. Osservando la Figura 4.4, che mostra la percentuale di errore media all'aumentare di $k$, si nota che dopo circa 135 iterazioni la performance

dell'algoritmo rimane stabile, perciò in questo caso saranno sufficienti circa $k = 130$–$140$ iterazioni per ottenere un clustering accurato.

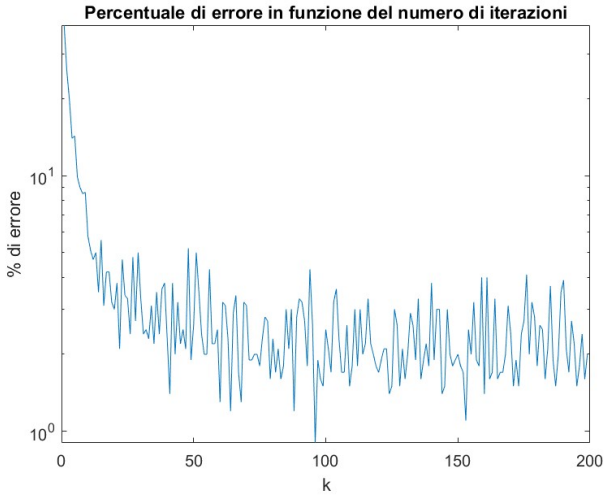


**Figura 4.4:** *Confronto percentuale di errore e tempo all'aumentare di k*

# Conclusioni

Una delle conclusioni più significative di questo elaborato è che la decomposizione CUR offre una soluzione valida a due importanti problemi nell'analisi dei dati: la fattorizzazione esatta delle matrici, descritta nel Teorema 2.1, che in questo caso può essere utilizzata per il problema del Subspace Clustering, ma anche l'approssimazione a rango basso di matrici. Inoltre, si ritiene che l'utilità della decomposizione CUR per il clustering e altre applicazioni potrebbe aumentare in futuro: una possibile direzione potrebbe essere quella di combinare la tecnica CUR con metodi sparsi per costruire algoritmi fortemente robusti al rumore e in grado di eseguire il clustering anche quando i punti dati non sono estratti da un'unione di sottospazi indipendenti. Di seguito, vengono fornite alcune ragioni che supportano l'utilità pratica delle decomposizioni CUR, specialmente in relazione all'analisi dei dati e al clustering.

Vantaggi della decomposizione CUR:

- Da un punto di vista teorico, la decomposizione CUR di una matrice utilizza una struttura di frame piuttosto che una struttura di base per fattorizzare la matrice, e pertanto gode di un livello di flessibilità superiore rispetto ad altri metodi come la SVD. Infatti, una base garantisce una rappresentazione unica con vettori linearmente indipendenti, perciò è minimale e priva di ridondanza, mentre un frame offre una rappresentazione potenzialmente non unica con vettori che possono essere ridondanti, e questo fornisce maggiori flessibilità e robustezza, soprattutto in presenza di rumore o perdita di informazioni.

- Una decomposizione CUR rimane fedele alla struttura dei dati. Ad esempio, se i dati iniziali sono sparsi, allora sia $C$ che $R$ saranno sparse, anche se $U^{\dagger}$ non lo è in generale. Al contrario, l'applicazione della SVD a una matrice sparsa genera, in generale, matrici $U$ e $V$ piene.

- Nella raccolta di dati reali, spesso molti elementi della matrice di dati possono mancare o essere estremamente corrotti. Nella segmentazione del moto, ad esempio, alcune caratteristiche potrebbero essere oscurate per diversi frame. Di conseguenza,

potrebbe essere necessario completare la matrice in qualche modo. D'altro canto, nella decomposizione CUR intere righe di una matrice di dati $W$ possono mancare, purché si possano comunque scegliere un numero sufficiente di righe in modo che la matrice risultante $R$ abbia lo stesso rango di $W$.

# Appendice A

# Codici MATLAB

## A.1 Funzione decomposizioneCUR

```
function[C,U,R]=decomposizioneCUR(A,M)

Input:
- A matrice di dati
- M numero atteso di sottospazi
Output:
- C sottoinsieme delle colonne di A
- R sottoinsieme delle righe di A
- U ''intersezione'' di U e R

[m,n]=size(A);
% s=M*4; k=n;     % per Hopkins155 (s=M*(dimensione attesa dei sottospazi))
s=M*3; k=n;       % per Database of Faces

selected_columns=sort(randperm(n,k));
C=A(:,selected_columns);

selected_rows=sort(randperm(m,s));
R=A(selected_rows,:);

U=A(selected_rows,selected_columns);
approx_A=C*pinv(U)*R;
return
```

## A.2 Funzione clusteringCUR

```
function[cidx,i,sigmamedia]=clusteringCUR(A,names,M,k,p,case1,case2)

Input:
- A matrice di dati su cui fare clustering sulle colonne
- names vettore contenente i nomi dei dati
- M numero atteso di sottospazi
- k numero massimo di iterazioni
- p esponente della matrice di similarità
- case1: 1=mediana, 2=media
- case2: 1=k-medie, 2=Spectral Clustering, 3=Principal Coordinate Clustering
Output:
- cidx matrice contenente i centroidi
- i numero di iterazioni effettuate
- sigmamedia matrice di similarità finale a cui viene applicato l'algoritmo
  di clustering

[m,n]=size(A);
matricisim=zeros(n);
i=1;
while i<k
    % calcolo decomposizione CUR
    [~,U,R]=decomposizioneCUR(A,M);
    Y=pinv(U)*R;
    % threshold
    Y=Y.*(abs(Y)>0.001);
    sigma=(Y'*Y)^p;
    % forzare connessioni note
    sigma=sigma-diag(diag(sigma))+eye(size(sigma));
    matricisim(:,:,i)=sigma;
    % controllo sul numero di iterazioni
    if i>1 && norm(matricisim(:,:,i)-matricisim(:,:,i-1))<0.01, break, end
    i=i+1;
end
```

```
if case1==1
    % mediana
    sigmamedia=abs(median(matricisim,3));
elseif case1==2
    % media
    sigmamedia=abs(mean(matricisim,3));
end
if case2==1
    % k-medie
    [cidx,~]=test1_kmeans(sigmamedia,names,M);
elseif case2==2
    % Spectral Clustering
    [cidx]=spectral_clustering(sigmamedia,names,M);
elseif case2==3
    % Principal Coordinate Clustering
    [cidx]=PCC(sigmamedia,names,M);
end
return
```

## A.3 Funzione spectral_clustering

```
function[cidx]=spectral_clustering(W,names,M)

Input:
- W matrice di similarità
- names vettore contenente i nomi dei dati
- M numero atteso di sottospazi
Output:
- cidx matrice contenente i centroidi

[m,n]=size(W);
% calcolo della matrice laplaciana
D=diag(W*ones(n,1));
L=D-W;
[autovett,autoval]=eig(L);
[~,idx]=sort(diag(autoval),'ascend');
```

```
autovett=autovett(:,idx);
% k_medie sugli M autovettori di L relativi agli autovalori più piccoli
[cidx,~]=test1_kmeans(autovett(:,1:M),names,M);
return
```

## A.4 Funzione PCC

```
function[cidx]=PCC(W,names,M)

Input:
- W matrice di similarità
- names vettore contenente i nomi dei dati
- M numero atteso di sottospazi
Output:
- cidx matrice contenente i centroidi

[~,ss,vv]=svd(W);
[cidx,~]=test1_kmeans((ss(1:M,1:M)*vv(:,1:M)')',names,M);
return
```

## A.5 CUR_hopkins

```
function[time,p_err,it]=CUR_hopkins(Data,k,p,case1,case2)

Input:
- Data structure array contenente
   - Data.width, Data.height dimensioni in pixels di tutti i frames
     nella sequenza video
   - Data.points numero di punti tracciati N
   - Data.frames numero di frames F
   - Data.y matrice 3xNxF contenente le coordinate omogenee dei punti
     tracciati in tutti i frames
   - Data.x matrice 3xNxF derivata da Data.y normalizzando le prime due
     componenti di ogni vettore per far sì che appartengano all'intervallo [-1,1]
   - Data.K matrice 3x3 di normalizzazione usata per passare da Data.y a Data.x
   - Data.s vettore Nx1 contenente la segmentazione corretta, cioè per ogni
```

```
     punto fornisce l'indice del corrispondente gruppo di moto
- k numero massimo di iterazioni per la decomposizione CUR
- p esponente della matrice di similarità
- case1: 1=mediana, 2=media
- case2: 1=k-medie, 2=Spectral Clustering, 3=Principal Coordinate Clustering
Output:
- time tempo medio generato da 10 runs dell'algoritmo CUR
- p_err percentuale di errore generata da 10 runs dell'algoritmo CUR
- it numero di iterazioni medio della decomposizione CUR in 10 runs

x=Data.x;
s=Data.s;
% eliminiamo la coordinata z perchè è sempre 1
x(3,:,:)=[];
[m,n,q]=size(x);
% riordiniamo gli elementi del tensore x nella matrice dei dati W
W=zeros(2*q,n);
for i=1:n
    for j=1:q
        W(2*j-1:2*j,i)=x(:,i,j);
    end
end
% ricaviamo il numero di sottospazi dal vettore s
M=max(s);
temp=zeros(10,1); p=zeros(10,1); num_iter=zeros(10,1);
for run=1:10
    tic
    [cidx,iter,~]=clusteringCUR(W,s,M,k,p,case1,case2);
    temp(run)=toc;
    num_iter(run)=iter;
    % in cidx ad ogni punto viene associato un indice del corrispondente
    % gruppo di moto, che può differire da quello associato allo stesso punto in
    % s. Tuttavia, il clustering risulta comunque corretto poichè il gruppo di
    % moto rimane lo stesso. Di conseguenza, per ottenere una percentuale di errore
    % attendibile creiamo delle corrispondenze tra gli indici in cidx e in s
    for p=1:M
```

```
        indici_s=find(s==p);
        indici_cidx=find(cidx==p);
        corrispondenza(p)=mode(s(indici_cidx));
    end
    for t=1:length(cidx)
        cidx(t)=corrispondenza(cidx(t));
    end
    p(run)=(length(find(cidx~=s))/length(s))*100;
end
time=mean(temp);
p_err=mean(p);
it=round(mean(num_iter));
return
```

## A.6 CUR_faces

```
function[time,p_err,it]=CUR_faces(Data,M,k,p,case1,case2)

Input:
- Data matrice contenente sulle colonne le 10 immagini degli M soggetti
- M numero atteso di sottospazi (corrispondente al numero di soggetti
  selezionati dal dataset)
- k numero massimo di iterazioni per la decomposizione CUR
- p esponente della matrice di similarità
- case1: 1=mediana, 2=media
- case2: 1=k-medie, 2=Spectral Clustering, 3=Principal Coordinate Clustering
Output:
- time tempo medio generato da 10 runs dell'algoritmo CUR
- p_err percentuale di errore generata da 10 runs dell'algoritmo CUR
- it numero di iterazioni medio della decomposizione CUR in 10 runs

% creiamo il vettore contenente il gruppo di appartenenza corretto per
% ogni immagine
names=[];
for q=1:M
    names=[names;repmat(q,10,1)];
```

```
end

temp=zeros(10,1); p=zeros(10,1); num_iter=zeros(10,1);
for run=1:10
    tic
    [cidx,iter,~]=clusteringCUR(Data,names,M,k,p,case1,case2);
    temp(run)=toc;
    num_iter(run)=iter;
    % calcoliamo la percentuale di errore come in CUR_hopkins
    for p=1:M
        indici_names=find(names==p);
        indici_cidx=find(cidx==p);
        corrispondenza(p)=mode(names(indici_cidx));
    end
    for t=1:length(cidx)
        cidx(t)=corrispondenza(cidx(t));
    end
    p(run)=(length(find(cidx~=names))/length(names))*100;
end
time=mean(temp);
p_err=mean(p);
it=round(mean(num_iter));
return
```

# Bibliografia


[1] A. Aldroubi, A. Sekmen, A.B. Koku, A.F. Cakmak: Similarity matrix framework for data from union of subspaces. *Applied and Computational Harmonic Analysis*, 45(2), 425-435 (2017).

[2] A. Sekmen, A. Aldroubi, A.B. Koku, K. Hamm: Principal Coordinate Clustering. *2017 IEEE International Conference on Big Data*, 2095-2101 (2017).

[3] A. Aldroubi, K. Hamm, A.B. Koku, A. Sekmen: CUR Decompositions, Similarity Matrices, and Subspace Clustering. *Frontiers in Applied Mathematics and Statistics*, 4:65 (2019).

[4] K. Hamm, L. Huang: Perspectives on CUR Decompositions. *Applied and Computational Harmonic Analysis*, 48(3), 1088-1099 (2020).

[5] M.W. Mahoney, P. Drineas: CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3), 697-702 (2009).

[6] V. Simoncini: Metodi Matriciali per il Data Science, Dispensa, Anno Accademico 2022/2023.

[7] P. Paramadevan, S. Sotheeswaran: Properties of Adjacency Matrix of a Graph and Its Construction. *Journal of Science EUSL*, 12(1), 13-21 (2021).

[8] S.L. Campbell, C.D. Meyer,Jr.: "Generalized Inverses of Linear Transformations". First edition. Dover Publications, Inc.: New York, 1991.

[9] B. Alexeev, J. Cahill, D.G. Mixon: Full spark frames. *Journal of Fourier Analysis and Applications*, 18(6), 1167-1194 (2012).

[10] J. Yan, M. Pollefeys: A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. *9th European Conference on Computer Vision*, 94-106 (2006).

[11] K. Kanatani, C.Matsunaga: Estimating the Number of Independent Motions for Multibody Motion Segmentation. *5th Asian Conference on Computer Vision*, 7-9 (2002).

[12] N.H. Barnouti, S.S.M. Al-Dabbagh, W.E. Matti: Face Recognition: A Literature Review. *International Journal of Applied Information Systems*, 11(4), 21-31 (2016).

[13] AT&T Laboratories Cambridge (1994). ORL Database of Faces disponibile all'url `https://cam-orl.co.uk/facedatabase.html`

[14] R. Tron, R. Vidal: A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. *IEEE International Conference on Computer Vision and Pattern Recognition*, 1-8 (2007). Dataset Hopkins155 disponibile all'url `http://www.vision.jhu.edu/data/hopkins155/`

[15] R. Vidal: A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2), 52-68 (2010).

# Ringraziamenti

Per prima cosa, vorrei rivolgere un sentito ringraziamento alla professoressa Valeria Simoncini. Sin dalle sue lezioni, è stata per me un modello di riferimento, trasmettendomi la passione per aspetti della matematica che non conoscevo e che ora apprezzo profondamente e spero di approfondire in futuro. La sua disponibilità, la sua gentilezza e il suo entusiasmo per la materia sono stati fondamentali per la stesura di questa tesi.

Ringrazio di cuore i miei genitori, Barbara e Massimo, per avermi permesso di fare questo percorso e per avermi sempre supportata incondizionatamente. Vi ringrazio perchè avete sempre creduto in me, soprattutto quando la mia eccessiva ansia cercava di convincermi che non ce l'avrei mai fatta, e per avermi ricordato di aver fiducia in me stessa ogni volta che ne avevo bisogno. Spero di avervi resi fieri di me. Un grazie di cuore va anche alla mia sorellina Margherita, la cui forza e determinazione sono per me fonte di ispirazione quotidiana, perchè so che ci sosteniamo sempre a vicenda, anche senza dirlo. Un ringraziamento speciale va ai miei nonni, Uliana e Uriano, e Fiorenza e Giorgio, la cui saggezza e amore hanno sempre guidato il mio cammino. Anche se alcuni di loro non sono più con noi, il loro ricordo e i loro insegnamenti continuano a vivere in me e mi hanno sostenuto durante tutto il percorso di studi. Un profondo grazie va al resto della mia famiglia, alle mie zie, ai miei zii e ai miei cugini, per avermi motivata a perseverare nei momenti di difficoltà e per aver sempre creduto nelle mie capacità.

Grazie a Sebastiano per avermi spronato ogni volta che ne avevo bisogno e per avermi trasmesso il tuo coraggio, e soprattutto per essere sempre stato al mio fianco senza mai dubitare di me.

Ringrazio le mie coinquiline Aurora, Grazia e Sofia, con le quali ci siamo sempre incoraggiate a vicenda, per avermi sempre fatta sentire a casa e per le tantissime risate.

Grazie ai miei compagni di corso per aver condiviso con me gioie e difficoltà di questo percorso, per aver reso le lezioni un po' più leggere e per tutti i pomeriggi di studio passati insieme.