Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Matematica

L'ALGORITMO DI ORTHOGONAL RANK-ONE MATRIX PURSUIT PER IL COMPLETAMENTO MATRICIALE

Tesi di Laurea in Analisi Numerica

Relatore:
Chiar.ma Prof.ssa
VALERIA SIMONCINI

Presentata da:
LISA PIZZO

Anno Accademico 2023/2024

Introduzione

Lo sviluppo e la comprensione di metodi che usano matrici di rango basso sono oggetto di crescente interesse nel machine learning grazie alle sue numerose applicazioni, tra cui il filtraggio collaborativo, il compressed sensing e il completamento delle matrici. In questo elaborato studiamo alcuni algoritmi per risolvere il problema del completamento matriciale: Data una matrice $Y \in \mathbb{R}^{n \times m}$ parzialmente osservata, cerchiamo una matrice $X \in \mathbb{R}^{n \times m}$ con rango minimo che abbia gli stessi elementi non nulli di Y. Poiché minimizzare il rango di una matrice è un problema computazionalmente difficile da controllare in maniera esatta, utilizziamo la norma nucleare della matrice. Tuttavia, minimizzare la norma nucleare di una matrice di grandi dimensioni risulta computazionalmente molto costoso in quanto richiede il calcolo dei vettori singolari. Negli ultimi anni, diversi matematici hanno lavorato per trovare soluzioni efficienti e accurate a questo problema, ma molti dei metodi attualmente conosciuti utilizzano ancora il calcolo completo della decomposizione in valori singolari, rendendoli non adatti a matrici di grandi dimensioni.

L'algoritmo studiato in questo elaborato necessita il calcolo della sola prima coppia di vettori singolari, rendendolo molto efficiente anche in caso di matrici di grandi dimensioni. L'algoritmo studiato estende l'algoritmo Orthogonal Matching Pursuit (OMP) dal caso vettoriale a quello matriciale. Ad ogni iterazione viene generata una base di matrici utilizzando la prima coppia di vettori singolari sinistri e destri della matrice residua. Successivamente, i coefficienti della combinazione lineare vengono aggiornati tramite una proiezione ortogonale della matrice di osservazione sul loro sottospazio generato.

Il calcolo della prima coppia di vettori singolari rimane computazionalmente costoso, richiedendo $\mathcal{O}(|\Omega|)$ flops ad ogni iterazione, dove Ω è l'insieme degli indici degli elementi non nulli della matrice di partenza Y parzialmente osservata.

Tuttavia, l'algoritmo ha una convergenza lineare e richiede solo $\mathcal{O}(\log(1/\epsilon))$ iterazioni per raggiungere una precisione ϵ.

Il principale problema dell'algoritmo studiato è l'elevato consumo di memoria, in quanto deve memorizzare $r|\Omega|$ elementi alla r-esima iterazione. Per risolvere questo problema, in questo elaborato viene anche studiato un secondo algoritmo chiamato EOR1MP, che richiede il calcolo, e quindi la memorizzazione, di sole due matrici per iterazione: la matrice calcolata nel passo precedente e la nuova matrice da aggiungere alla base. Pertanto, quando le matrici sono ristrette ai soli elementi di Ω, è necessario solo $2|\Omega|$ di spazio di memoria per ogni passo. Nonostante questa ottimizzazione, l'algoritmo mantiene una convergenza lineare.

Nel primo capitolo richiameremo definizioni e concetti utili per comprendere l'elaborato.

Nel secondo capitolo esamineremo l'algoritmo OR1MP, dimostrando la sua efficacia computazionale per il completamento matriciale e la sua convergenza lineare.

Nel terzo capitolo, analizzeremo la sua versione ottimizzata in termini di memoria, l'algoritmo EOR1MP, e ne dimostreremo la convergenza lineare.

Infine, nel quarto capitolo, applicheremo l'algoritmo OR1MP a due casi di studio e confronteremo i risultati ottenuti con quelli di altri algoritmi esistenti.

Indice

Capitolo 1

Prerequisiti

1.1 Richiami

Ricordiamo alcune definizioni che risulteranno utili nel seguito dell'elaborato.

Definizione 1.1. Una matrice $A \in \mathbb{R}^{n\times n}$ si dice ortogonale se $A^TA = AA^T = I_n$ dove I_n indica la matrice identità $n \times n$.

Definizione 1.2. Una funzione $|| \cdot || : \mathbb{C}^{m\times n} \to \mathbb{R}$ è una norma di matrice se sono soddisfatte le seguenti proprietà:

1. $||A|| \geq 0$ per ogni $A \in \mathbb{C}^{m\times n}$ e $||A|| = 0$ se e solo se $A = 0$
2. $||\alpha A|| = |\alpha|\ ||A||$, per ogni $\alpha \in \mathbb{C},\ A \in \mathbb{C}^{m\times n}$
3. $||A + B|| \leq ||A|| + ||B||$, per ogni $A, B \in \mathbb{C}^{m\times n}$
4. $||AB|| \leq ||A||\ ||B||$, per ogni $A \in \mathbb{C}^{m\times n},\ B \in \mathbb{C}^{n\times p}$

Esempio (Norme)**.** Sia $Y = (Y_{ij})_{i=1,\dots,n;\ j=1,\dots,m} \in \mathbb{C}^{n\times m}$:

- Norma di Frobenius
$$||Y||_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} |Y_{ij}|^2\right)^{1/2}$$

- Norma-2
$$\|Y\|_2 = \max_{x \neq 0} \frac{\|Yx\|_2}{\|x\|_2}, x \in \mathbb{C}^m$$
dove $\|x\|_2 = (\sum_{i=1}^{m} |x_i|^2)^{1/2}$.

1.2 Notazioni

Nel presente elaborato useremo le seguenti notazioni:

- $Y = [y_1 \dots y_m]$ è una matrice $\mathbb{R}^{n \times m}$ scritta per colonne. $\Omega \subseteq \{1 \dots n\} \times \{1 \dots m\}$ è l'insieme di indici relativi a valori non nulli della matrice Y;
- $P_\Omega(Y)$, o brevemente Y_Ω, è la proiezione di Y sullo spazio generato dalle matrici che hanno elementi non nulli solo negli indici di Ω. In altre parole la componente (i, j)-esima di $P_\Omega(Y)$ è uguale a $Y_{i,j}$ per ogni $(i, j) \in \Omega$ ed è zero altrimenti;
- $\text{vec}(Y) = (y_1^T, \dots, y_m^T)^T$ è l'operazione di concatenazione di tutte le colonne in un unico vettore. Mentre $\dot{y} = \text{vec}(P_\Omega(Y))$ è la concatenazione dei soli valori non nulli di Y;
- $\langle X, Y \rangle = \langle \text{vec}(X), \text{vec}(Y) \rangle = \text{trace}(X^T Y)$ dove $\langle \cdot, \cdot \rangle$ indica il prodotto interno di due vettori;
- $\|Y\| = \sqrt{\langle Y, Y \rangle} = \|Y\|_F$
- $\|\theta\|_0$, chiamata norma l_0, è il numero di elementi non nulli di θ

1.3 Decomposizione in valori singolari

La decomposizione in valori singolari è un tipo di decomposizione matriciale che permette di estendere il concetto di diagonalizzazione anche per matrici rettangolari, permettendo di prendere due matrici ortogonali diverse come matrici di trasformazione. In questa sezione ne sarà data la definizione e saranno fornite le prime proprietà fondamentali, si veda la sezione 3 del capitolo 5 di [3].

Teorema 1.3.1. Sia $A \in \mathbb{R}^{m\times n}$ e supponiamo senza perdere di generalità $m \geq n$. Allora esistono $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m\times m}$, $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n\times n}$, e $\Sigma \in \mathbb{R}^{m\times n}$, tali che:

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

dove U, V sono matrici ortogonali e Σ è una matrice diagonale con $\sigma_1, \ldots, \sigma_n$ sulla diagonale tali che $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Dimostrazione. Sia $A \in \mathbb{R}^{m\times n}$ con $m \geq n$. Dimostriamo che esistono tali U, Σ, V. Se A è la matrice nulla allora il risultato è banale, sia quindi A la matrice non nulla e sia $x = \arg\max_{\|\tilde{x}\|=1} \|A\tilde{x}\|_2$. Sia $Ax = \sigma_1 y$ con $\|y\|_2 = 1$, dove $\sigma_1 = \|A\|$. Definiamo ora $X_1 = [x, \tilde{X}_2] \in \mathbb{R}^{n\times n}$ e $Y_1 = [y, \tilde{Y}_2] \in \mathbb{R}^{m\times m}$ in modo che siano entrambe ortogonali. Allora, visto che $y^T Ax = \sigma_1$ e $\tilde{Y}_2 Ax = \sigma_1 \tilde{Y}_2^T y = 0$, si ha

$$A_1 = Y_1^T A X_1 = \begin{bmatrix} \sigma_1 & d^T \\ 0 & B \end{bmatrix}$$

Inoltre si dimostra che $d = 0$, infatti posto $z = \begin{bmatrix} \sigma_1 \\ d \end{bmatrix}$ si ha

$$\begin{aligned} \sigma_1^2 = \|A\|^2 = \|A_1\|^2 \geq \frac{\|A_1 z\|^2}{\|z\|^2} &= \frac{\left\| \begin{bmatrix} \sigma_1 & d^T \\ 0 & B \end{bmatrix} \begin{bmatrix} \sigma_1 \\ d \end{bmatrix} \right\|^2}{\sigma_1^2 + d^T d} \\ &= \frac{(\sigma_1^2 + d^T d)^2 + \|Bd\|^2}{\sigma_1^2 + d^T d} \\ &\geq \sigma_1^2 + d^T d = \sigma_1^2 + \|d\|^2, \end{aligned}$$

da cui $\sigma_1^2 \geq \sigma_1^2 + \|d\|^2$ se e solo se $\|d\|^2 = 0$ ossia $d = 0$. Abbiamo quindi che

$$A_1 = Y_1^T A X_1 = \begin{bmatrix} \sigma_1 & 0 \\ 0 & B \end{bmatrix}.$$

Si procede ora in modo iterativo su B fino a trovare $U = Y_1 \cdot \ldots \cdot Y_{n-1}$, $V = X_1 \cdot \ldots \cdot X_{m-1}$

□

I vettori $u_1, \ldots, u_m$ sono detti vettori singolari sinistri, i vettori $v_1, \ldots, v_n$ sono detti vettori singolari destri, e $\sigma_1, \ldots, \sigma_n$ sono detti valori singolari.

Se per $k \in \{1 \ldots n-1\}$ si ha $\sigma_{k+1} \ll \sigma_k$, allora

$$A = \sum_{i=1}^{n} u_i \sigma_1 v_i^T = \sum_{i=1}^{k} u_i \sigma_i v_i^T + \sum_{i=k+1}^{n} u_i \sigma_i v_i^T \approx \sum_{i=1}^{k} u_i \sigma_i v_i^T = A_k.$$

Teorema 1.3.2.

$$A_k = \arg \min_{\substack{B \in \mathbb{R}^{m \times n} \\ \text{rango}(B)=k}} \|A - A_k\|_2 = \sigma_{k+1}$$

Dimostrazione. Sia $k = \text{rango}(A_k)$ e $\|A - A_k\|_2 = \|\sum_{i>k} u_i \sigma_i v_i^T\|_2 = \sigma_{k+1}$. Sia ora B di rango k e sia $\{x_1, \ldots, x_{n-k}\}$ base dello spazio nullo di B. Allora dev'essere $Z = \{x_1, \ldots, x_{n-k}\} \cap \{v_1, \ldots, v_{k+1}\} \neq \{0\}$ poiché al massimo si possono avere n vettori linearmente indipendenti. Sia $z \in Z$, $\|z\| = 1$. Allora $Bz = 0$ e $Az = \sum_{i=1}^{k+1} u_i \sigma_i v_i^T z$. Quindi:

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

e A_k raggiunge il minimo. □

Definizione 1.3. Definiamo ora la norma nucleare di una matrice $A \in \mathbb{R}^{n \times m}$ come

$$\|A\|_1 = \sum_{i=1}^{r} \sigma_i$$

dove σ_i sono i valori singolari di A e r è il rango della matrice A.

1.4 Orthogonal Matching Pursuit

I risultati di questa sezione descrivono il contenuto di [4].

L'algoritmo OMP, acronimo di Orthogonal Matching Pursuit, è un algoritmo che permette di risolvere il seguente problema: Dato un vettore $b \in \mathbb{R}^m$ e una matrice $A \in \mathbb{R}^{m \times n}$ tale che $m \ll n$, trovare $x \in \mathbb{R}^m$ con pochi elementi non nulli tale che $Ax = b$. L'idea dell'algoritmo è quella di trovare $x \in \mathbb{R}^m$ con pochi elementi non nulli sfruttando il fatto che b è una combinazione lineare sparsa delle colonne di A. L'algoritmo è un

algoritmo iterativo: ad ogni iterazione prima identifica quale colonna di A ha la maggiore correlazione con il vettore b, l'indice di questa colonna sarà anche l'indice di un elemento non nullo di x che verrà determinato risolvendo il problema ai minimi quadrati, dopodiché itera il ragionamento con le colonne rimaste.

Termini e definizioni utili:

- **Supporto**: Sia $x \in \mathbb{R}^m$, l'insieme degli indici degli elementi non nulli di x è chiamato supporto, ossia $\Lambda = \{i : x_i \neq 0\}$
- **Sparsità**: La sparsità di un vettore x è il numero di elementi non nulli, ossia $|\Lambda(x)|$ oppure $\|x\|_0$
- **Correlazione normalizzata**: Per i vettori $\{x_1, x_2, \ldots, x_n\}$ con $x_j \in \mathbb{R}^m, \forall j$ definiamo la correlazione come

$$\rho_{x_k, x_j} = \frac{\langle x_k, x_j \rangle}{\|x_k\|_2 \|x_j\|_2}.$$

L'algoritmo costruisce il supporto Λ in modo iterativo aggiungendo l'indice corrispondente ad una colonna selezionato opportunamente ad ogni iterazione. In particolare, dato un vettore b, una matrice A, l'algoritmo inizializza il supporto come l'insieme vuoto e il residuo come il vettore b. Il primo passo dell'algoritmo consiste nel determinare la colonna di A con maggiore correlazione con il vettore residuo r_{k-1} e tale colonna la si determina nel seguente modo

$$\lambda_k = \arg \max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle|.$$

Notiamo che cerchiamo una colonna di indice $j \notin \Lambda_{k-1}$ per evitare di estrarre una colonna già utilizzata precedentemente. Notiamo inoltre che è importante eliminare eventuali colonne doppie di A prima di avviare l'algoritmo, infatti se ci fossero colonne doppie $\arg\max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle|$ avrebbe più di una soluzione.

Il secondo passo dell'algoritmo è quello di aggiornare il supporto, ossia $\Lambda_k = \Lambda_{k-1} \cup \{\lambda_k\}$. Il terzo passo consiste nel determinare x_k attraverso la risoluzione di un problema ai minimi quadrati

$$\begin{cases} (x_k)_i = \arg\min_x \|A_{\Lambda_k} x - b\|_2 & \text{se } i \in \Lambda_k \\ (x_k)_i = 0 & \text{se } i \notin \Lambda_k \end{cases},$$

dove $(x_k)_i$ indica l'elemento i-esimo del vettore x determinato alla k-esima iterazione e A_{Λ_k} è la sottomatrice di A che ha come colonne solo quelle di indici inclusi in Λ_k. Il quarto passo consiste nel calcolare $\hat{b}_k = Ax_k$ ossia consiste nel determinare l'approssimazione di b usando una combinazione lineare tra le colonne di A e i coefficienti di x_k alla k-esima iterazione. Infine, l'algoritmo calcola il residuo $r_{k+1} = b - \hat{b}_k$.

L'algoritmo è quindi riassunto così

Algorithm Orthogonal Matching Pursuit (OMP)

Input: la matrice A, il vettore noto b

Inizializzazione: $\Lambda_0 = \emptyset$, $r_0 = b$

Normalizzare tutte le colonne di A rispetto alla norma unitaria $\| \circ \|_2$

Rimuovere le colonne duplicate di A

for $k = 1, 2, \ldots$ **do**

$\lambda_k = \arg\max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle|$

$\Lambda_k = \Lambda_{k-1} \cup \{\lambda_k\}$

$(x_k)_i = \arg\min_x \|A_{\Lambda_k} x - b\|_2$ per $i \in \Lambda_k$

$\hat{b}_k = Ax_k$

$r_k = b - \hat{b}_k$

end

Output: una rappresentazione sparsa x ed il relativo supporto Λ_k

Capitolo 2

Orthogonal Rank-One Matrix Pursuit

I risultati di questo capitolo descrivono il contenuto di [1]

2.1 Completamento matriciale di rango basso

Sia $Y \in \mathbb{R}^{n \times m}$ una matrice osservata parzialmente, ossia con molti elementi non noti. Il completamento matriciale consiste nel trovare una matrice $X \in \mathbb{R}^{n \times m}$ di rango minimo tale che $P_\Omega(X) = P_\Omega(Y)$, dove Ω è l'insieme degli indici degli elementi non nulli della matrice Y e P_Ω è la proiezione di Y sul suo sottospazio generato. Per risolvere questo problema si minimizza la norma nucleare della matrice cercata.

Una matrice $X \in \mathbb{R}^{n \times m}$ può essere scritta come combinazione lineare di matrici di rango uno,

$$X = \sum_{i \in I} (\theta)_i M_i = M(\theta) \tag{2.1}$$

dove $\{M_i : i \in I\}$, chiamato dizionario, è l'insieme di tutte le matrici di rango 1 di dimensione $n \times m$ con norma di Frobenius unitaria e $(\theta) = ((\theta)_1, (\theta)_2, \dots) \in \mathbb{R}^{|I|}$ è il vettore dei coefficienti.

Il problema di approssimazione di una matrice di rango minimo consiste nel minimizzare la norma l_0 del vettore θ sotto alcune condizioni, ossia

$$\begin{aligned} &\min_{\theta} \|\theta\|_0 \\ &\text{t.c. } P_\Omega(M(\theta)) = P_\Omega(Y) \end{aligned} \tag{2.2}$$

Posto in altri termini, il problema che stiamo cercando di risolvere è il seguente

$$\begin{aligned} &\min_{\theta} \|P_\Omega(M(\theta)) - P_\Omega(Y)\|_F^2 \\ &\text{t.c. } \|\theta\|_0 \leq r. \end{aligned} \tag{2.3}$$

É possibile risolvere questo problema di minimo vincolato mediante un algoritmo simile all'algoritmo Orthogonal Matching Pursuit usando come base le matrici di rango uno.

Notiamo che non è possibile ricondurre il problema ad un problema già noto in quanto il dizionario $\{M_i : i \in I\}$ ha dimensione infinita e la base è costruita durante il processo di risoluzione del problema stesso e non è nota a priori. L'algoritmo studiato in questo elaborato cerca una base sovracompleta di matrici di rango uno e determina poi i coefficienti della combinazione lineare.

2.2 L'algoritmo

L'algoritmo è costituito da due passi.

Dopo la $k-1$-esima iterazione, supponiamo di avere la base $\{M_1, \ldots, M_{k-1}\}$ e il vettore delle coordinate (θ_{k-1}).

La correlazione tra due matrici $A, B \in \mathbb{R}^{n\times m}$ è definita come il prodotto interno normalizzato tra A e B.

Allora alla k-esima iterazione:

Passo uno: Cerchiamo una nuova matrice M_k di norma di Frobenius unitaria e rango 1, tale che sia maggiormente correlata al residuo $R_k = P_\Omega(Y) - X_{k-1}$ con $X_{k-1} = P_\Omega(M(\theta_{k-1})) = \sum_{i=1}^{k-1}(\theta_{k-1})_i P_\Omega(M_i)$ dove $(\theta_{k-1})_i$ è da intendersi come la i-esima componente del vettore θ determinato alla $k-1$-esima iterazione. Quindi M_k può essere calcolata come la soluzione del seguente problema

$$\max_M \{\langle M, R_k\rangle : \text{rango}(M) = 1,\ \|M\|_F = 1\}. \tag{2.4}$$

Scrivendo $M = uv^T$ con $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$ e $\|u\| = \|v\| = 1$, il problema (2.4) diventa

$$\begin{aligned} \langle M, R_k \rangle &= \text{trace}(M^T R_k) \\ &= \text{trace}(vu^T R) \\ &= u^T R_k v, \end{aligned} \tag{2.5}$$

dove è stata usata la proprietà di ciclicità della traccia.

Il problema che vogliamo risolvere diventa

$$\max_{u,v} \{u^T R_k v : \|u\| = \|v\| = 1\}. \tag{2.6}$$

Scomponiamo ora la matrice R_k in valori singolari, e otteniamo così

$$R_k = U \Sigma V^T,$$

dove U è una matrice ortogonale $n \times n$, Σ è una matrice diagonale $n \times m$ con i valori singolari in ordine decrescente sulla diagonale e V è una matrice ortogonale $m \times m$. Sostituendo ora la scomposizione di R_k in $u^T R_k v$ si ottiene

$$u^T R_k v = u^T U \Sigma V^T v.$$

Notiamo che, posti $u' = U^T u$ e $v' = V^T v$, $u' \in \mathbb{R}^n, v' \in \mathbb{R}^m$ sono ancora vettori di norma unitaria. Otteniamo così

$$\begin{aligned} u^T R_k v &= u'^T \Sigma v' \\ &= \sum_{i=1}^{\min(n,m)} \sigma_i u'_i v'_i. \end{aligned}$$

Risolvere il problema 2.4, ossia massimizzare $u^T R_k v$ equivale quindi a massimizzare $u'^T \Sigma v'$. Ricordando però che i vettori u', v' sono di norma unitaria, si deduce che il problema dipende dai valori singolari σ_i posti in ordine decrescente sulla diagonale della matrice Σ. Il valore massimo è dunque σ_1, per cui il problema 2.4 è risolto dalla coppia di vettori singolari destri e sinistri relativa al primo valore singolare della matrice R_k, ossia (u_1, v_1) determinanti senza calcolare l'intera decomposizione in valori singolari ma sfruttando la decomposizione troncata. In conclusione la nuova matrice M_k di rango uno da aggiungere alla base sarà la matrice $M_k = u_1 v_1^T$.

Passo due: Per aggiornare il vettore θ_k, vettore dei coefficienti della nuova base $\{M_1, \ldots, M_k\}$, è sufficiente risolvere il seguente problema ai minimi quadrati

$$\min_{\theta \in \mathbb{R}^k} \Big\| \sum_{i=1}^{k} \theta_i M_i - Y \Big\|_\Omega^2. \tag{2.7}$$

Se riscriviamo le matrici $(Y)_\Omega$ e $(M_i)_\Omega$ nei vettori $\dot{y}$ e $\dot{m}_i$, allora la soluzione θ_k del problema (2.7) è data da

$$\theta_k = \arg\min_\theta \|y - \overline{M}_k(\theta)\| \tag{2.8}$$

dove $\overline{M}_k = [\dot{m}_1, \ldots, \dot{m}_k]$ è la matrice formata da tutte le matrici della base, riscritte in forma vettoriale. Il numero delle righe della matrice $\overline{M}_k$ sarà quindi il numero totale degli elementi diversi da zero.

L'algoritmo è costituito dai due passi appena illustrati, e vengono ripetuti fino a quando sarà soddisfatto uno dei criteri di arresto. Il metodo può essere terminato quando la matrice raggiunge un rango scelto oppure quando il residuo $\|R_k\|$ è minore di una data tolleranza ϵ.

L'algoritmo OR1MP è riassunto qui sotto:

Algorithm 1 Orthogonal Rank-One Matrix Pursuit (OR1MP)

Input: Y_Ω e le soglie per il criterio d'arresto scelto

Inizializzazione: $X_0 = 0$ e $k = 1$

repeat

Step 1: Trovare la coppia principale di vettori singolari sinistri e destri (u_k, v_k) del residuo osservato $R_k = Y_\Omega - X_{k-1}$ e definire $M_k = u_k(v_k)^T$

Step 2: Calcolare i coefficienti θ_k risolvendo il problema ai minimi quadrati ossia $\theta_k = \arg\min_\theta \|y - \overline{M}(\theta)\|$

Step 3: Porre $X_k = \sum_{i=1}^{k} (\theta_k)_i (M_i)_\Omega$ e $k = k + 1$

until sono soddisfatti i criteri di arresto

Output: La matrice stimata $\hat{Y} = \sum_{i=1}^{k} (\theta_k)_i M_i$

2.3 Analisi di convergenza dell'algoritmo OR1MP

Dimostriamo ora che l'algoritmo studiato converge linearmente. A tal fine richiamiamo innanzi tutto la definizione di convergenza lineare.

Definizione 2.1. Un metodo converge linearmente alla soluzione $\bar{x}$ se

$$\lim_{k\to+\infty} \frac{|x_{k+1} - \bar{x}|}{|x_k - \bar{x}|} = \mu$$

con $\mu \in (0, 1)$.

Il risultato principale è dato dal seguente teorema

Teorema 2.3.1. Posto $X_{k-1} = \sum_{i=1}^{k-1} (\theta_{k-1})_i (M_i)_\Omega$, il residuo $R_k = Y_\Omega - X_{k-1}$ dell'algoritmo orthogonal rank-one matrix pursuit soddisfa

$$\|R_k\| \leq \left(\sqrt{1 - \frac{1}{\min(m, n)}}\right)^{k-1} \|Y\|_\Omega, \ \forall k \geq 1.$$

Ma prima di dimostrare il Teorema 2.3.1, abbiamo bisogno di dimostrare alcune proprietà preliminari. Prima di tutto dimostriamo che la formula (2.8) è ben definita e che θ_k è univocamente determinato prima che l'algoritmo si interrompa. Per fare questo ci servono le seguenti proprietà. La prima proprietà dice che la matrice del residuo R_{k+1} è perpendicolare alle matrici M_i della base precedentemente generate $i = 1, \ldots, k$.

Proprietà 2.3.2. $\langle R_{k+1}, M_i \rangle = 0$ per $i = 1, \ldots, k$.

Dimostrazione. Essendo questo un problema ai minimi quadrati, è noto che il residuo vec(R) è ortogonale a Range$(\overline{M}_k)$. Si può quindi concludere che il residuo calcolato alla iterazione $k + 1$ è ortogonale a tutte le colonne della matrice $\overline{M}_k$ e quindi ortogonale a tutte le k matrici della base, ossia la tesi. □

La seguente proprietà mostra che mentre il numero di matrici di rango uno della base cresce durante l'iterazione, il residuo decresce in modo debolmente monotono.

Proprietà 2.3.3. $\|R_{k+1}\| \leq \|R_k\|$ per ogni $k \geq 1$.

Dimostrazione. Osserviamo che per ogni $k \geq 1$,

$$\begin{aligned}\|R_{k+1}\|^2 &= \min_{\theta\in\mathbb{R}^k}\{\|Y - \sum_{i=1}^{k}\theta_i M_i\|_\Omega^2\} \\ &\leq \min_{\theta\in\mathbb{R}^k,\theta_k=0}\{\|Y - \sum_{i=1}^{k}\theta_i M_i\|_\Omega^2\} \\ &= \min_{\theta\in\mathbb{R}^{k-1}}\{\|Y - \sum_{i=1}^{k-1}\theta_i M_i\|_\Omega^2\} = \|R_k\|^2\end{aligned}$$

da cui la tesi. □

La terza proprietà dimostra che $\{(M_i)_\Omega\}_{i=1}^k$ è linearmente indipendente a meno che $\|R_k\| = 0$.

Proprietà 2.3.4. Supponiamo che $R_k \neq 0$ per qualche $k \geq 1$. Allora, $\overline{M}_i$ ha rango per colonne massimo per tutti gli $i \leq k$.

Dimostrazione. Ricordiamo che R_k è la matrice del residuo alla k-esima iterazione e che (u_1, v_1) è la prima coppia di vettori singolari sinistri e destri della matrice R_k con i quali definiamo $M_k = u_1 v_1^T$. Ricordiamo inoltre che $\overline{M}_k = [\dot{m}_1, \ldots, \dot{m}_k]$ e quindi ogni nuova colonna di $\overline{M}_k$ è definita a partire dai vettori singolari del residuo precedente. Per la proprietà 2.3.3 e l'ipotesi $R_k \neq 0$ per qualche $k \geq 1$, si ha che $R_i \neq 0$ per ogni $i \leq k$. Dimostriamo ora la tesi procedendo per induzione su i.
Passo base: Se $R_1 \neq 0$ allora è ovvio avere $\overline{M}_1 \neq 0$.
Passo induttivo: Supponiamo l'ipotesi induttiva, ossia supponiamo che valga per $i-1 < k$ e dimostriamolo per $i \leq k$. Per ipotesi induttiva $\overline{M}_{i-1}$ è di rango massimo, supponiamo per assurdo che $\overline{M}_i$ non abbia rango massimo, allora esiste $\alpha \in \mathbb{R}^{i-1}$ tale che

$$(M_i)_\Omega = \sum_{j=1}^{i-1}\alpha_j (M_j)_\Omega.$$

Per la proprietà 2.3.2 si ha che $\langle R_i, M_i\rangle = 0$, Ne segue che

$$\sigma_{\max}(R_i) = u_1^T R_i v_1 = \langle R_i, M_i\rangle = 0$$

da cui $R_i = 0$, che contraddice l'ipotesi che $R_j \neq 0$ per ogni $j \leq k$. Quindi $\overline{M}_i$ è di rango massimo, da cui la tesi. □

Per poter dimostrare la convergenza lineare serve ora stabilire una relazione tra $\|R_{k+1}\|$ e $\|R_k\|$.

Poniamo per semplicità $(\theta_{k-1})_k = 0$ e $\theta_k = \theta_{k-1} + \eta_k$ dove

$$\eta_k = \arg\min_{\eta} \|\overline{M}_k(\eta) - R_k\|_\Omega^2. \tag{2.9}$$

dove $(\theta_{k-1})_k$ è la componente k-esima del vettore θ determinato alla $k-1$-esima iterazione. Definiamo ora L_k tale che

$$\text{vec}(L_k) = (\overline{M}_k)_\Omega(\eta_k) \tag{2.10}$$

Dalla definizione di $X_k = \sum_{i=1}^{k}(\theta_k)_i(M_i)_\Omega$ è inoltre possibile osservare che $X_k = X_{k-1} + L_k$ e $R_{k+1} = R_k - L_k$.

Proprietà 2.3.5. $\|R_{k+1}\|^2 = \|R_k\|^2 - \|L_k\|^2$ e $\|L_k\|^2 \geq \langle M_k, R_k\rangle^2$, dove L_k è definita in 2.10.

Dimostrazione. Sappiamo $R_{k+1} = R_k - \overline{M}_k(\eta)$ da cui $R_{k+1}^T\overline{M}_k = 0$. Otteniamo così

$$\begin{aligned} R_k^T R_k &= (R_{k+1} + \overline{M}_k(\eta))^T(R_{k+1} + \overline{M}_k(\eta)) \\ &= R_{k+1}^T R_{k+1} + (\overline{M}_k(\eta))^T(\overline{M}_k(\eta)) \end{aligned}$$

da cui

$$\begin{aligned} \|R_{k+1}\|^2 &= \|R_k\|^2 - \|\overline{M}_k(\eta)\|^2 \\ &= \|R_k\|^2 - \|L_k\|^2. \end{aligned}$$

Inoltre, se $R_k = 0$ allora è ovvio che $\|L_k\|^2 \geq \langle M_k, R_k\rangle^2$. Supponiamo quindi che $R_k \neq 0$. Segue allora dalla proprietà 2.3.4 che $\overline{M}_k$ ha rango massimo e quindi usando anche l'equazione (2.9) si ottiene $\eta_k = (\overline{M}_k^T\overline{M}_k)^{-1}\overline{M}_k^T\dot{r}_k$ dove $\dot{r}_k = \text{vec}(R_k)_\Omega$. Otteniamo che

$$\|L_k\|^2 = \dot{r}_k^T\overline{M}_k(\overline{M}_k^T\overline{M}_k)^{-1}\overline{M}_k^T\dot{r}_k.$$

Sia ora $\overline{M}_k = QU$ la fattorizzazione QR della matrice $\overline{M}_k$ dove Q è una matrice ortogonale, ossia $Q^TQ = I$ e U è una matrice $k \times k$ triangolare superiore e non singolare. Possiamo osservare che $(\overline{M}_k)_k = \dot{m}_k$, dove $(\overline{M}_k)_k$ è la k-esima colonna della matrice $\overline{M}_k$ e $\dot{m}_k$ è la matrice $(\overline{M}_k)_\Omega$ scritta vettorialmente. Ricordando che $\|M_k\| = \|u_k v_k^T\| = 1$ si ha $\|(\overline{M}_k)_k\| \leq 1$ e siccome $Q^TQ = I$ e $\overline{M}_k = QU$ si ha che, per definizione di U,

$$0 \leq |U_{kk}| \leq \|U_k\| = \|(\overline{M}_k)_k\| \leq 1.$$

Usando anche la proprietà 2.3.2 si ottiene

$$\overline{M}_k^T \dot{r}_k = [0, \ldots, 0, \langle M_k, R_k \rangle]^T.$$

Da cui, siccome U è triangolare superiore e tale che $|U_{k,k}| \leq 1$, si ha

$$\begin{aligned}
\|L_k\|^2 &= \dot{r}_k^T \overline{M}_k (\overline{M}_k^T \overline{M}_k)^{-1} \overline{M}_k^T \dot{r}_k \\
&= \dot{r}_k^T \overline{M}_k (U^T U)^{-1} \overline{M}_k^T \dot{r}_k \\
&= [0, \ldots, 0, \langle M_k, R_k \rangle] U^{-1} U^{-T} [0, \ldots, 0, \langle M_k, R_k \rangle]^T \\
&= \frac{\langle M_k, R_k \rangle^2}{(U_{k,k})^2} \geq \langle M_k, R_k \rangle^2
\end{aligned}$$

□

Dimostrazione del Teorema 2.3.1. Per $M_k = u_k (v_k)^T$ si ha

$$\begin{aligned}
\langle M_k, R_k \rangle &= \langle u_k (v_k)^T, R_k \rangle = \sigma_1(R_k) \\
&= \|R_k\|_2 \geq \frac{1}{\sqrt{\min(n,m)}} \|R_k\|_F.
\end{aligned}$$

dove $\sigma_1(R_k)$ è il valore singolare massimo della matrice del residuo R_k. Usando la proprietà 2.3.5 otteniamo

$$\begin{aligned}
\|R_{k+1}\|^2 &= \|R_k\|^2 - \|L_k\|^2 \\
&\leq \|R_k\|^2 - \langle M_k, R_k \rangle^2 \\
&\leq \big(1 - \frac{1}{\min(m,n)}\big) \|R_k\|^2
\end{aligned}$$

Siccome $\|R_1\| = \|Y\|_\Omega^2$, possiamo concludere che

$$\|R_k\| \leq \left(\sqrt{1 - \frac{1}{\min(m,n)}}\right)^{k-1} \|Y\|_\Omega$$

□

Capitolo 3

Economic Orthogonal Rank-One Matrix Pursuit

I risultati di questo capitolo descrivono il contenuto di [2].

3.1 L'algoritmo

L'algoritmo OR1MP precedentemente proposto deve allocare in memoria tutte le basi calcolate, necessita quindi $O(r|\Omega|)$ spazio di memoria per ottenere una matrice di rango r. Per problemi di grandi dimensioni, non è possibile sottovalutare lo spazio di memoria necessario, risulta quindi necessario abbassare il rango della matrice stimata.

Per adattare l'algoritmo a problemi di grandi dimensioni e per permettere alla matrice stimata di avere un rango sufficientemente alto, è stato semplificato il secondo passo dell'iterazione. Sarà quindi sufficiente calcolare, e quindi memorizzare, la matrice stimata nell'iterazione $k-1$ e la nuova matrice di norma unitaria M_k. In questo caso è sufficiente stimare solo i due coefficienti delle due matrici, e lo si fa risolvendo il seguente problema ai minimi quadrati

$$\alpha_k = \arg \min_{\alpha=\{\alpha_1,\alpha_2\}} \|(\alpha_k)_1 X_{k-1} + (\alpha_k)_2 M_k - Y\|_\Omega^2, \tag{3.1}$$

dove $\alpha_k \in \mathbb{R}^2$ è il coefficiente stimato all'iterazione k, e $(\alpha_k)_i$ con $i = 1, 2$ è invece la i-esima componente del vettore α_k.

Se scriviamo la matrice stimata come combinazione lineare della base, otteniamo $X_k = \sum_{i=1}^{k} (\theta_k)_i (M_i)_\Omega$ con $(\theta_k)_k = (\alpha_k)_2$ e $(\theta_k)_i = (\theta_{k-1})_i \cdot (\alpha_k)_1$ per $i < k$, dove $(\theta_{k-1})_i$ è la i-esima componente del vettore θ determinato alla $(k-1)$-esima iterazione e $(\alpha_k)_1$ è la prima componente del vettore α determinato alla k-esima iterazione.

L'algoritmo economico EOR1MP è riassunto qui sotto:

Algorithm 2 Economic Orthogonal Rank-One Matrix Pursuit (EOR1MP)

Input: Y_Ω e le soglie per il criterio d'arresto scelto

Inizializzazione: $X_0 = 0$ e k=1

repeat

Step 1: Trovare una coppia di vettori singolari sinistri e destri (u_k, v_k) del residuo osservato $R_k = Y_\Omega - X_{k-1}$ e definire $M_k = u_k (v_k)^T$

Step 2: Calcolare i coefficienti α_k per X_{k-1} e M_k risolvendo $\arg\min_\alpha \|(\alpha_k)_1 X_{k-1} + (\alpha_k)_2 (M_k)_\Omega - Y_\Omega\|^2$

Step 3: Porre $X_k = (\alpha_k)_1 X_{k-1} + (\alpha_k)_2 (M_k)_\Omega$; $(\theta_k)_k = (\alpha_k)_2$ e $(\theta_k)_i = (\theta_{k-1})_i (\alpha_k)_1$ per $i < k$ e $k = k+1$

until sono soddisfatti i criteri di arresto

Output: La matrice stimata $\hat{Y} = \sum_{i=1}^{k} (\theta_k)_i M_i$

3.2 Analisi di convergenza dell'algoritmo EOR1MP

L'algoritmo EOR1MP occupa decisamente meno spazio di memoria dell'algoritmo precedente OR1MP, ma nonostante questo l'algoritmo EOR1MP converge comunque linearmente.

Dimostriamo la convergenza lineare dimostrando il seguente teorema.

Teorema 3.2.1. Posto $X_{k-1} = (\alpha_{k-1})_1 X_{k-1} + (\alpha_{k-1})_2 (M_{k-1})_\Omega$ con $(\theta_{k-1})_{k-1} = (\alpha_{k-1})_2$ e $(\theta_{k-1})_i = (\theta_{k-2})_i \cdot (\alpha_{k-1})_1$. Il residuo $R_k = Y_\Omega - X_{k-1}$ dell'algoritmo economic orthogonal rank-one matrix pursuit soddisfa

$$\|R_k\| \leq \left(\sqrt{1 - \frac{1}{\min(m,n)}} \right)^{k-1} \|Y\|_\Omega, \ \forall k \geq 1.$$

Ma prima di dimostrare il teorema 3.2.1, è necessario dimostrare diverse importanti proprietà. La prima proprietà dice che R_{k+1} è perpendicolare alle matrici X_{k-1} e M_k.

Proprietà 3.2.2. $\langle R_{k+1}, X_{k-1}\rangle = 0$ e $\langle R_{k+1}M_k\rangle = 0$.

Dimostrazione. Essendo un problema ai minimi quadrati, è noto che il residuo vec(R_{k+1}) è ortogonale a Range$(\overline{M}_k)$ da cui $\langle R_{k+1,}M_k\rangle = 0$. Siccome $X_{k-1} = \sum_{i=1}^{k-1}(\theta_k)_i, (M_i)_\Omega$ si ha

$$\begin{aligned}\langle R_{k+1}, X_{k-1}\rangle &= \langle R_{k+1}, \sum_{i=1}^{k-1}(\theta_k)_i(M_i)_\Omega\rangle \\ &= \sum_{i=1}^{k-1}(\theta_k)_i\langle R_{k+1}, (M_i)_\Omega\rangle = 0\end{aligned}$$

□

Proprietà 3.2.3. $\|R_{k+1}\|^2 = \|Y_\Omega\|^2 - \|X_k\|^2$ per ogni $k \geq 1$.

Dimostrazione. Dalla definizione di $R_k = Y_\Omega - X_{k-1}$ e dalla 3.2.2 la tesi è immediata. □

La seguente proprietà mostra che mentre il numero di matrici di rango uno della base cresce, il residuo $\|R_k\|$ decresce in modo debolmente monotono.

Proprietà 3.2.4. $\|R_{k+1}\| \leq \|R_k\|$ per ogni $k \geq 1$.

Dimostrazione. Osserviamo che per ogni $k \geq 1$,

$$\begin{aligned}\|R_k\|^2 &= \|Y - ((\alpha_{k-1})_1 X_{k-2} + (\alpha_{k-1})_2 M_{k-1})\|_\Omega^2 \\ &\geq \min_{\alpha\in\mathbb{R}^2} \|Y - \alpha_1((\alpha_{k-1})_1 X_{k-2} + (\alpha_{k-1})_2 M_{k-1}) - \alpha_2 M_k\|_\Omega^2 \\ &= \min_{\alpha\in\mathbb{R}^2} \|Y - \alpha_1 X_{k-1} - \alpha_2 M_k\|_\Omega^2 \\ &= \|Y - (\alpha_{k+1})_1 X_{k-1} - (\alpha_{k+1})_2 M_k\|_\Omega^2 \\ &= \|R_{k+1}\|^2\end{aligned}$$

da cui la tesi. □

Definiamo ora

$$A_k = B_k^T B_k = \begin{bmatrix} \langle X_{k-1}, X_{k-1}\rangle & \langle X_{k-1}, M_k\rangle \\ \langle M_k, X_{k-1}\rangle & \langle M_k, M_k\rangle_\Omega \end{bmatrix}$$

con $B_k = \Big[\mathrm{vec}(X_{k-1}), \mathrm{vec}((M_k)_\Omega)\Big]$. Sappiamo che la soluzione del problema (3.1) in aritmetica esatta è

$$\alpha_k = A_k^{-1} B_k^T \mathrm{vec}(Y_\Omega).$$

Per dimostrare ciò, riscriviamo il problema (3.1) in forma vettoriale. Otteniamo così $\mathrm{vec}((\alpha_k)_1 X_{k-1} + (\alpha_k)_2 M_k) = (\alpha_k)_1 \mathrm{vec}(X_{k-1}) + (\alpha_k)_2 \mathrm{vec}(M_k)_\Omega$ e $\mathrm{vec}(Y)_\Omega = \dot{y}$. Allora il problema diventa

$$\alpha_k = \arg\min_{\alpha\in\mathbb{R}^2} \|B_k\alpha - \dot{y}\|_\Omega^2.$$

Per l'equazione normale, la soluzione analitica a questo problema è quindi

$$\alpha_k = (B_k^T B_k)^{-1} B_k^T \dot{y} = A_k^{-1} B_k^T \dot{y}.$$

Dimostriamo che $\mathrm{vec}(X_{k-1})$ e $\mathrm{vec}((M_k)_\Omega)$ sono linearmente indipendenti se $\|R_k\| \neq 0$, da cui segue che A_k è invertibile e α_k è univocamente determinato prima che l'algoritmo si interrompa.

Proprietà 3.2.5. Se $X_{k-1} = \beta \cdot (M_k)_\Omega$ per un qualche $\beta \neq 0$ allora $\|R_{k+1}\| = \|R_k\|$

Dimostrazione. Se $X_{k-1} = \beta \cdot (M_k)_\Omega$ con $\beta \neq 0$ allora si ha

$$\begin{aligned}
\|R_{k+1}\|^2 &= \min_{\alpha\in\mathbb{R}^2} \|Y - \alpha_1 X_{k-1} - \alpha_2 M_k\|_\Omega^2 \\
&= \min_{\alpha\in\mathbb{R}^2} \|Y - (\alpha_1 + \alpha_2/\beta) X_{k-1}\|_\Omega^2 \\
&= \min_{\gamma\in\mathbb{R}} \|Y - \gamma X_{k-1}\|_\Omega^2 \\
&= \min_{\gamma\in\mathbb{R}} \|Y - \gamma(\alpha_{k-1})_1 X_{k-2} - \gamma(\alpha_{k-1})_2 M_{k-1}\|_\Omega^2 \\
&\geq \min_{\gamma\in\mathbb{R}^2} \|Y - \gamma_1 X_{k-2} - \gamma_2 M_{k-1}\|_\Omega^2 \\
&= \|Y - X_{k-1}\|_\Omega^2 \\
&= \|R_k\|^2.
\end{aligned}$$

Riassumendo si ha $\|R_{k+1}\|^2 \geq \|R_k\|^2$. Da questo e dalla proprietà 3.2.4 si ha la tesi. □

Proprietà 3.2.6. $\langle M_k, R_k\rangle \geq \frac{1}{\sqrt{\min(m,n)}}\|R_k\|$ per ogni $k \geq 1$.

Dimostrazione. $\langle M_k, R_k\rangle = \sigma_1(R_k) = \|R_k\|_2 \geq \frac{1}{\sqrt{\min(m,n)}}\|R_k\|_F$ dove $\sigma_1(R_k)$ è il massimo valor singolare di R_k. □

Proprietà 3.2.7. Sia $R_k \neq 0$ per qualche $k \geq 1$. Allora $X_{k-1} \neq \beta \cdot (M_k)_\Omega$ per ogni $\beta \neq 0$.

Dimostrazione. Nella proprietà 3.2.5 abbiamo dimostrato che se $X_{k-1} = \beta \cdot (M_k)_\Omega$ con $\beta \neq 0$ allora si ha $\|R_{k+1}\|^2 = \|R_k\|^2$ dove $\|R_k\|^2 = \|Y - X_{k-1}\|_\Omega^2$ e

$$\begin{aligned}
\|R_{k+1}\|^2 &= \|Y - X_k\|_\Omega^2 \\
&= \min_{\alpha\in\mathbb{R}^2} \|Y - \alpha_1 X_{k-1} - \alpha_2 M_k\|_\Omega^2 \\
&= \min_{\alpha\in\mathbb{R}^2} \|Y - (\alpha_1 + \alpha_2/\beta) X_{k-1}\|_\Omega^2 \\
&= \min_{\gamma\in\mathbb{R}} \|Y - \gamma X_{k-1}\|_\Omega^2 \\
&= \|Y - \gamma_k X_{k-1}\|_\Omega^2.
\end{aligned}$$

È quindi evidente che $\gamma^k = 1$ è l'unica soluzione da cui $\langle X_{k-1}, R_k\rangle = 0$. Questo è però in contraddizione con

$$\langle X_{k-1}, R_k\rangle = \beta\langle M_k, R_k\rangle = \beta\sigma_1(R_k) \neq 0.$$

da cui la tesi. □

La seguente proprietà costruisce una relazione tra due residui consecutivi $\|R_{k+1}\|$ e $\|R_k\|$.

Proprietà 3.2.8. $\|R_{k+1}\|^2 \leq \|R_k\|^2 - \frac{\sigma_1^2(R_k)}{\langle M_k, M_k\rangle_\Omega}$.

Dimostrazione.

$$\begin{aligned}
\|R_{k+1}\|^2 &= \min_{\alpha\in\mathbb{R}^2} \|Y - \alpha_1 X_{k-1} - \alpha_2 M_k\|_\Omega^2 \\
&\leq \min_{\alpha_2\in\mathbb{R}} \|Y - X_{k-1} - \alpha_2 M_k\|_\Omega^2 \\
&= \min_{\alpha_2\in\mathbb{R}} \|R_k - \alpha_2 M_k\|_\Omega^2.
\end{aligned}$$

La soluzione analitica al precedente problema è $\alpha_2^* = \frac{\langle R_k, M_k \rangle}{\langle M_k, M_k \rangle_\Omega}$. Sostituendo questo dentro l'equazione precedente troviamo

$$\begin{aligned}\|R_{k+1}\|^2 &\leq \|R_k - \frac{\langle R_k, M_k \rangle}{\langle M_k, M_k \rangle_\Omega} M_k\|_\Omega^2 \\ &= \|R_k\|^2 - \frac{\langle R_k, M_k \rangle^2}{\langle M_k, M_k \rangle_\Omega} \\ &= \|R_k\|^2 - \frac{\sigma_1^2(R_k)}{\langle M_k, M_k \rangle_\Omega}\end{aligned}$$

dove $\sigma_1(R_k)$ è il primo valore singolare della matrice R_k. □

Dimostrazione del teorema 3.2.1. Siccome Ω non contiene tutti gli indici allora si ha $\langle M_k, M_k \rangle_\Omega \leq 1$. Per le proprietà 3.2.8 e 3.2.6 si ha che

$$\begin{aligned}\|R_{k+1}\|^2 &\leq \|R_k\|^2 - \frac{\sigma_1^2(R_k)}{\langle M_k, M_k \rangle_\Omega} \\ &\leq \|R_k\|^2 - \sigma_1^2(R_k) \\ &\leq \left(1 - \frac{1}{\min(m,n)}\right)\|R_k\|^2.\end{aligned}$$

E, per $\|R_1\| = \|Y\|_\Omega^2$, si ottiene

$$\|R_k\| \leq \left(\sqrt{1 - \frac{1}{\min(m,n)}}\right)^{k-1} \|Y\|_\Omega.$$

□

Capitolo 4

Applicazioni

In questo capitolo applichiamo l'algoritmo OR1MP alla risoluzione di vari problemi e lo confrontiamo con due altri algoritmi: Singular Value Projectiong (SVP)[6] e Singular Value Thresholding (SVT)[7]. I codici di questi ultimi sono disponibili al seguente link: `https://github.com/HauLiang/Matrix-Completion-Methods`. Gli esperimenti sono stati eseguiti su un MacBook Air 2020 con macOS Sonoma 14.3, dotato di chip Apple M1 con 8 core (4 prestazioni e 4 efficienza) e 8 GB di RAM. Gli esperimenti sono stati condotti in MATLAB®.

4.1 Recupero di immagini

Per questo esperimento è stato utilizzato il database "Yale Faces", scaricato dal seguente link: `http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html`. Il database contiene 165 immagini in formato GIF di 15 individui diversi, con 11 immagini per persona, rappresentante una diversa espressione facciale: centrale, con gli occhiali, felice, con la luce da sinistra, senza occhiali, normale, con la luce da destra, triste, stanco, sorpreso e con l'occhiolino. Per l'esperimento, sono state utilizzate le seguenti 5 immagini:

- Individuo 2 - espressione 7
- Individuo 6 - espressione 10

- Individuo 10 - espressione 3
- Individuo 13 - espressione 5
- Individuo 15 - espressione 11

Per semplicità, le foto sono denominate "numero dell'individuo.espressione". Quindi, le 5 foto selezionate sono indicate come $2.7, 6.10, 10.3, 13.5, 15.11$.

Da ogni foto viene rimosso circa il 50% dei pixel, scelti casualmente utilizzando il comando `randi` di Matlab. Successivamente, l'algoritmo OR1MP viene applicato alla matrice sparsa, ovvero l'immagine contenente solo il 50% dei pixel originali. L'algoritmo si ferma quando soddisfa il criterio di arresto impostato, che in questo caso è $\|R_k\| < 10^{-4}\|R_0\|$.

L'algoritmo, come scritto in MATLAB® è riassunto qui sotto:

```
C=imread('2.7','jpeg');
C1=rgb2gray(C);
[dim1,dim2]=size(C1);
omega=randi([0,1],dim1,dim2,1);
Cmod=omega.*double(C1);
[Y_hat,iter] = R1MP(Cmod,omega);

------------------------------------------------------------

function [Y_hat,iter] = R1MP(Y,Omega)
[n,m]=size(Y); idx=find(Omega>0); doty=Y(idx);
k=1; Theta=0; R=Y;
Matrix=[]; %Serve per tenere memoria di tutte le matrici che
    costituiscono la base
OverlineM=[]; %Serve per calcolare Theta
while norm(R)/norm(Y)>10e-4
    %Step 1
```

```
17      [u,~,v]=svds(R,1); %Solo la prima coppia di vettori
            singolari
18      M=u*v'; %Nuova matrice della base
19      dotm=M(idx);
20      Matrix=[Matrix,reshape(M,n*m,1)];
21      OverlineM=[OverlineM,dotm];
22      %Step 2
23      Theta=OverlineM\doty;
24      %Step 3
25      X=zeros(n,m);
26      X=reshape(Matrix*Theta,[n, m]).* Omega;
27      R=Y-X;
28      k=k+1;
29  end
30  iter=k-1;
31  Y_hat=reshape(Matrix(:,1:k-1)*Theta(1:k-1),[n, m]);
32  end
```

Si presentano ora i risultati ottenuti:

Per prima cosa, viene messa a confronto l'efficacia dei tre algoritmi analizzati tramite il confronto delle immagini ricostruite con l'immagine originale.

Figura 4.1: *Confronto dell'immagine 6.10 originale con le immagini ricostruite utilizzando i tre algoritmi esaminati.*

È evidente che l'algoritmo OR1MP è il più efficace nel generare un'immagine molto simile a quella iniziale. Tra gli altri due algoritmi considerati, SVT risulta essere il migliore.

Successivamente, i tre algoritmi vengono eseguiti per 10 volte ciascuno, e grazie ai comandi *tic* e *toc* viene registrato il tempo impiegato per completare la matrice.

La tabella seguente riporta la media del numero di iterazioni e il tempo medio impiegato dagli algoritmi per completare la matrice.

Per problemi legati alla potenza del computer utilizzato, il criterio di arresto degli algoritmi è stato modificato nel seguente $\|R_k\| < 10^{-3}\|R_0\|$.

Immagine	OR1MP		SVP		SVT	
	Iterazioni	Tempo	Iterazioni	Tempo	Iterazioni	Tempo
2.7	32.0	1.2663 s	156	2.2410 s	842	18.3243 s
6.10	36.5	1.4353 s	56	0.8347 s	675	11.1719 s
10.3	34.9	1.4915 s	113	1.8525 s	519	9.1384 s
13.5	22.9	0.7821 s	169	2.4492 s	536	8.7905 s
15.11	27.3	1.1644 s	88	1.4577 s	516	9.2873 s

Tabella 4.1: Risultati della ricostruzione di immagini rappresentati in termini di iterazioni e tempo impiegato dall'algoritmo per soddisfare il criterio di arresto.

Si può quindi concludere che per quanto l'efficacia dell'algoritmo SVT sia comparabile a quella dell'algoritmo OR1MP, il tempo di esecuzione di SVT è significativamente più alto rispetto a quello di OR1MP. In altre parole, l'algoritmo studiato in questo elaborato risulta essere il migliore quando si considerano sia la precisione che il tempo di esecuzione.

In conclusione, viene mostrata la convergenza lineare dell'algoritmo.

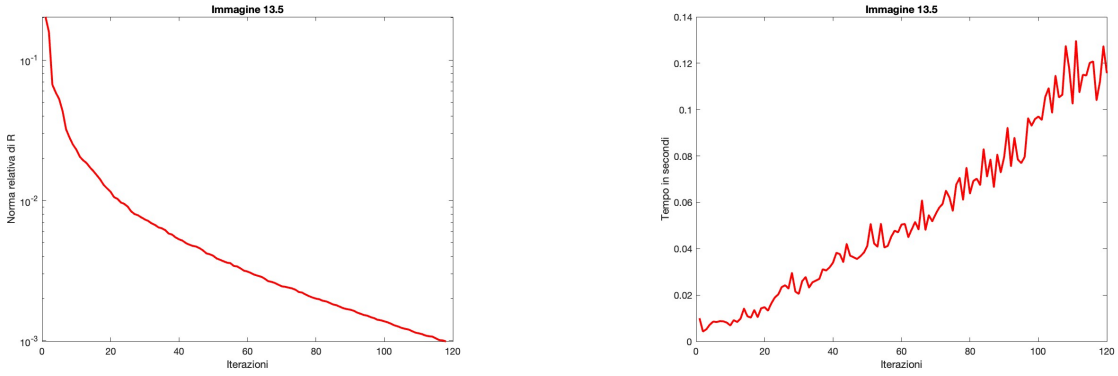

Figura 4.2: *Convergenza dell'algoritmo OR1MP sull'immagine 13.5: sull'asse x è riportato il rango, sull'asse y la norma di R (immagine a sinistra) e il tempo di esecuzione misurato in secondi (immagine a destra).*

4.2 Raccomandazioni

Per il secondo esperimento sono stati utilizzati due diversi dataset di grandi dimensioni: Jester3, MovieLens100K, MovieLens1M. I data set sono scaricabili dai seguenti link: `https://grouplens.org/datasets/movielens/` e `https://eigentaste.berkeley.edu/dataset/`.

Il dataset Jester contiene valutazioni su 100 diverse barzellette da quasi 25 mila persone diverse. I voti sono numeri reali compresi tra -10.00 e 10.00, e quando non è stato espresso un voto, è stato riportato il valore 99.

Il dataset MovieLens è una collezioni di valutazioni fatte dagli utenti del sito MovieLens, con voti che vanno da 1 a 5.

Le dimensioni esatte dei tre datasets usati sono rappresentati nella seguente tabella.

Dataset	# rows	# column	# rating
Jester3	24938	100	6×10^5
MovieLens100K	943	1682	10^5
MovieLens1M	6040	3952	10^6

Tabella 4.2: Dimensioni dei dataset utilizzati.

L'algoritmo OR1MP è confrontato con gli algoritmi SVP e SVT già usati in precedenza.

Per l'esperimento, è stato rimosso circa il 50% dei dati dalla tabella iniziale, creando così due matrici distinte. Gli algoritmi sono stati applicati alla matrice contenente solo il 50% dei dati, mentre la seconda matrice, contenente il 100% dei dati originalmente noti, è stata utilizzata per verificare l'accuratezza delle previsioni degli algoritmi.

Il criterio di arresto utilizzato è $\|R_k\| < 10^{-2}\|R_0\|$.

L'algoritmo, come scritto su MATLAB® è riassunto qui sotto:

```
T = readtable("MovieLens1M.xlsx");
i=table2array(T(:,1)); %Indici di riga
j=table2array(T(:,2)); %Indici di colonna
v=table2array(T(:,3)); %Valori dei voti espressi
n=6040; m=3952; %Dati per il dataset MovieLens1M
```

```
C=sparse(i,j,v,n,m);
null=randi([0,1],length(v),1,1);
i_masked=nonzeros(i.*null);
j_masked=nonzeros(j.*null);
v_masked=nonzeros(v.*null);
omega=sparse(i_masked,j_masked,1,n,m);
C_masked=sparse(i_masked,j_masked,v_masked,n,m);
[Y_hat,iter] = R1MP(C_masked,omega);
-------------------------------------------------------------
function [Y_hat,iter] = R1MP(Y,Omega)
[n,m]=size(Y); idx=find(Omega==1); doty=Y(idx);
k=1; Theta=0; R=Y;Matrix=[]; OverlineM=[];
while normest(R)/normest(Y)>10e-2
    %Step 1
    [u,~,v]=svds(R,1); %Prima coppia di vettori singolari
    M=u*v';
    dotm=M(idx);
    Matrix=[Matrix,reshape(M,n*m,1)];
    OverlineM=[OverlineM,dotm];
    %Step 2
    Theta=OverlineM\doty;
    %Step 3
    X=zeros(n,m);
    X=reshape(Matrix*Theta,[n, m]).* Omega;
    R=Y-X;
    k=k+1;
end
iter=k-1;
Y_hat=reshape(Matrix(:,1:k-1)*Theta(1:k-1),[n, m]);
end
```

Si presentano ora i risultati ottenuti:

I tre algoritmi vengono eseguiti 5 volte ciascuno, e il tempo impiegato per completare la matrice è registrato con i comandi *tic* e *toc*.

La tabella seguente riporta la media del numero di iterazioni e il tempo medio impiegato dagli algoritmi per soddisfare il criterio di arresto.

DataSet	**OR1MP**		**SVP**		**SVT**	
	Iter.	**Tempo**	**Iter.**	**Tempo**	**Iter.**	**Tempo**
Jester3	1.0	0.5028 s	3.0	0.4947 s	80.0	8.6704 s
MovieLens100K	13.8	0.4187 s	5.0	1.8337 s	16.0	10.1349 s
MovieLens1M	16.4	19.9626 s	5.0	118.3316 s	200	- s

Tabella 4.3: Risultati della raccomandazioni rappresentati in termini di iterazioni e tempo impiegato dagli algoritmi per soddisfare il criterio di arresto.

Si può notare che per il datare Jester3 gli algoritmi OR1MP e SVP presentano tempi di esecuzione comparabili mentre l'algoritmo SVT impiega più tempo. È invece evidente che per i due dataset di MovieLens, l'algoritmo studiato OR1MP è decisamente più efficiente in termini di tempo, soprattuto su MovieLens1M dove SVT raggiunge il numero di iterazioni massime prima di convergere per via del calcolo completo della decomposizione in valori singolari della matrice di dimensioni 10^6.

Successivamente, è stata calcolata la norma relativa dell'errore tra la matrice risultante dai due algoritmi e la matrice sparsa contenente il 100% dei dati noti. In questo modo, è possibile valutare l'efficacia dei due metodi.

La tabella seguente riporta le medie relative.

Dataset	**OR1MP**	**SVP**	**SVT**
Jester3	0.2453	0.3618	1.5186
MovieLens100K	0.5060	0.7013	0.3830
MovieLens1M	0.5016	0.8020	-

Tabella 4.4: Norma relativa tra la matrice sparsa con tutti i dati e la matrice ricostruita con gli algoritmi.

È possibile osservare che l'algoritmo OR1MP presenta in tutti e 3 i casi una norma relativa inferiore rispetto all'algoritmo SVP mentre l'algoritmo SVT presenta una norma inferiore a quella dell'algoritmo OR1MP nel caso del dataset MovieLens100K.

Unendo i due risultati è però evidente che l'algoritmo OR1MP è più efficace se si tiene conto del tempo impiegato e dell'efficacia nella previsione delle raccomandazioni.

Conclusioni

In questo elaborato è stato studiato un algoritmo per il completamento matriciale, efficace anche per matrici di grandi dimensioni. L'idea di base è estendere l'algoritmo Orthogonal Matching Pursuit, adattandolo dal caso vettoriale a quello matriciale.

Abbiamo esaminato l'algoritmo proposto in [1], insieme alla sua convergenza lineare. Tuttavia, abbiamo osservato che tale algoritmo richiedeva un eccessivo utilizzo di memoria, soprattutto all'aumentare delle dimensioni dei dati. Pertanto, è stato introdotto un secondo algoritmo, studiato in [2], più economico in termini di memoria. Quest'ultimo mantiene comunque la convergenza lineare.

Infine, l'algoritmo studiato è stato applicato a due differenti casi di studio di dimensioni diverse. In entrambi i casi, è stato illustrato che l'algoritmo studiato risulta essere il migliore, considerando sia l'efficienza che il tempo impiegato.

Bibliografia

[1] Z. Wang, M. Lai, Z. Lu, W. Fan, H. Davulcu, J. Ye, "Rank-One Matrix Pursuit for Matrix Completion", *Proceedings of the 31st International Conference on Machine Learning*: 91–99, (2014);

[2] Z.Wang, M.Lai, Z. Lu, W. Fan, H. Davulcu, J. Ye, "Orthogonal Rank-One Matrix Pursuit for Low Rank Matrix Completion", *ArXiv*, abs/1404.1377, (2014);

[3] D. Palitta e V. Simoncini, "Dispense del corso di Calcolo Numerico - Modulo di Algebra Lineare Numerica", II edizione, v.6, Università di Bologna, (2021);

[4] A. Ang, "Orthogonal Matching Pursuit Algorithm", *Department of Combinatorics and Optimization*, Waterloo, Canada, (2022);

[5] G.H. Golub e C.F. Van Loan, "Matrix Computations", III edizione, The Johns Hopkins University Press, (1996);

[6] P. Jain, R. Meka e I.S. Dhillon "Guaranteed Rank Minimization via Singular Value Projection", *Neural Information Processing Systems*, (2009).

[7] E. Candès e B. Recht, "Exact matrix completion via convex optimization", *Foundation of Computational Mathematics*, 9(6):717-772, (2009).