Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

DECOMPOSIZIONE CUR PER UNA MIGLIORE ANALISI DEI DATI

Tesi di Laurea in Analisi Numerica

Relatore:
Chiar.ma Prof.ssa
VALERIA SIMONCINI

Presentata da:
ANDREA SEBASTIANI

II Sessione
Anno Accademico 2016/2017

Ai miei genitori

"Due strade trovai nel bosco ed io scelsi quella meno battuta, ed è per questo che sono diverso."
Robert Frost

Indice

Introduzione

Le tecniche comunemente utilizzate per l'analisi dei dati sono la PCA (Principal component analysis) e la SVD (decomposizione ai valori singolari), e permettono di esprimere una matrice di dati (dataset) in termini di un insieme di vettori ortogonali ordinati in base alla loro importanza. Sfortunatamente, tali vettori, essendo combinazioni lineari di tutti i punti del dataset, risultano difficili da interpretare in termini del fenomeno/processo da cui provengono i dati o in termini dei dati stessi.
In questa tesi verrà presentata una particolare decomposizione matriciale di rango basso, la *decomposizione CUR*, che permette di rappresentare la matrice dei dati in termini di alcune righe e/o colonne della matrice stessa. Tale proprietà renderà più facile l'interpretazione dei risultati in termini dei dati di partenza.
Mediante tale decomposizione si cerca di approssimare una data matrice A come prodotto di tre matrici:

- C formata da alcune colonne di A
- R formata da alcune righe di A
- U costruita appositamente per rendere il prodotto CUR vicino ad A

Saranno presentati due algoritmi randomizzati che prendendo in input una matrice A restituiscono alcune delle sue colonne, per costruire la matrice C, ed alcune delle sue righe, per costruire la matrice R. La tecnica utilizzata dagli algoritmi per scegliere le colonne e le righe è un particolare strategia di sampling chiamata *subspace sampling*.
Nel primo capitolo, verranno presentati dei richiami di algebra lineare e dei risultati, riguardanti la SVD, che saranno utilizzati nel seguito.
Nel secondo capitolo, verranno presentate la decomposizione CX e la decomposizione CUR e i relativi algoritmi di subspace sampling per la costruzione delle matrici C ed R.
Nel terzo capitolo, verranno presentati i risultati dell'utilizzo della decomposizione CUR su una matrice termini-documento ricavata da una directory web.

Capitolo 1

Nozioni preliminari

1.1 Richiami di Algebra Lineare

Diamo alcune definizioni che poi utilizzeremo all'interno di questa tesi.

Definizione 1.1. Sia $A \in \mathbb{R}^{n\times n}$. Allora A è *ortogonale* se

$$AA^T = A^T A = I_n$$

Se A è ortogonale le sue colonne formano una base ortonormale di $\mathbb{R}^n$.
In particolare, se $A \in \mathbb{C}^{n\times n}$, si dice che A *unitaria* se

$$AA^* = A^* A = I_n$$

dove A^* è la trasposta coniugata di A.
Se A è ortogonale allora è anche unitaria.

Definizione 1.2. Data $A \in \mathbb{R}^{m\times n}$, si dice che A è *sparsa* se per ogni riga di A, gli elementi non nulli sono il $3-5\%$. In altre parole $a_{ij} = 0$ per la maggior parte degli indici $i = 1, \ldots, m$ e $j = 1, \ldots, n$.
Se A non è sparsa allora è detta *piena*.

Definizione 1.3. Sia $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$. Si definisce la *norma-2* di x come

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Definizione 1.4. Sia $A \in \mathbb{R}^{m\times n}$. Si definisce la *norma di Frobenius* di A come

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

La seguente norma è anche detta *norma euclidea indotta* poiché è definita utilizzando la norma euclidea vettoriale.

Definizione 1.5. Sia $A \in \mathbb{R}^{m\times n}$. Si definisce la *norma-2* di A come

$$\|A\|_2 = \sup_{\substack{x\in\mathbb{R}^n \\ \|x\|_2=1}} \|Ax\|_2 = \max_{0\neq x\in\mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2}$$

1.2 SVD completa

La decomposizione ai valori singolari o Singular Value Decomposition (SVD), è una fattorizzazione che permette di rappresentare una matrice data A come prodotto di tre matrici, di cui due ortogonali ed una diagonale

Teorema 1.1 (Esistenza della SVD)**.** *Una qualunque matrice $A \in \mathbb{R}^{m\times n}$ con $m \geq n$ può essere scritta come*

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T$$

dove $U \in \mathbb{R}^{m\times m}$ e $V \in \mathbb{R}^{n\times n}$ sono matrici ortogonali e $\Sigma \in \mathbb{R}^{n\times n}$ è diagonale

$$\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix}$$

con $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ detti valori singolari.
Le colonne di $U = [u_1, \ldots, u_m]$ sono dette vettori singolari sinistri mentre quelle di $V = [v_1, \ldots, v_n]$ vettori singolari destri.
La Singular Value Decomposition di A permette di scrivere la matrice come $A = \sum_{i=1}^{n} u_i\sigma_i v_i^T$ dove $u_i\sigma_i v_i^T$ prende il nome di i-esima componente singolare di A.

Osservazione. Sia Rango$(A) = r$ allora si ha $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ e $\sigma_{r+1} = \cdots = \sigma_n = 0$

Dimostrazione. Si osservi che non è restrittivo supporre $m \geq n$, se così non fosse basterebbe applicare il teorema ad A^T.
Si consideri il problema di minimo

$$\sup_{\|x\|_2=1} \|Ax\|_2$$

La soluzione di questo problema esiste perché si sta cercando il massimo di una funzione continua su un insieme chiuso, quindi il massimo è raggiunto per qualche vettore x. Sia $\bar{x}$ un vettore soluzione e sia

$$A\bar{x} = \sigma_1 y$$

dove $\|y\|_2 = 1$ e $\sigma_1 = \|A\|_2$. Si costruiscano quindi

$$X_1 = [\bar{x}, \tilde{X}_2] \in \mathbb{R}^{n\times n}, \qquad Y_1 = [y, \tilde{Y}_2] \in \mathbb{R}^{m\times m}$$

in modo che siano entrambi ortogonali. Quindi si ha

$$A_1 := Y_1^T A X_1 = \begin{pmatrix} \sigma_1 & y^T A\tilde{X}_2 \\ 0 & \tilde{Y}_2{}^T A\tilde{X}_2 \end{pmatrix} = \begin{pmatrix} \sigma_1 & d^T \\ 0 & B \end{pmatrix}$$

dato che $y^T A\bar{x} = \sigma_1$ e $\tilde{Y}_2{}^T A\bar{x} = \sigma_1 \tilde{Y}_2{}^T y = 0$.
Si osserva che

$$\frac{\|Ax\|^2}{\|x\|^2} = \frac{\|A_1 x\|^2}{\|x\|^2} \qquad \forall\, x \in \mathbb{R}^n$$

In quanto A_1 è ottenuta da A attraverso trasformazioni ortogonali che non cambiano la norma. Sia

$$x = \begin{pmatrix} \sigma_1 \\ d \end{pmatrix} \in \mathbb{R}^n$$

con $d \in \mathbb{R}^{n-1}$. Allora

$$\frac{\|Ax\|^2}{\|x\|^2} = \frac{\|A_1[\sigma_1, d]^T\|^2}{\|[\sigma_1, d]^T\|^2} = \frac{\|[\sigma_1^2 + d^T d, Bd]^T\|^2}{\sigma_1^2 + d^T d} =$$
$$= \frac{(\sigma_1^2 + d^T d)^2) + \|Bd\|^2}{\sigma_1^2 + d^T d} \geq \sigma_1^2 + d^T d$$

Quindi

$$\frac{\|Ax\|^2}{\|x\|^2} \geq \sigma_1^2 + d^T d$$

Essendo $\sigma_1 = \max_{0\neq x\in\mathbb{R}^n} \frac{\|Ax\|}{\|x\|}$ deve essere $d = 0$.
Quindi la prima riga e la prima colonna di A_1 sono zero, eccetto che per l'elemento diagonale σ_1. Iterando la procedura su B alla fine si avranno $U = Y_1 \cdots Y_{n-1}$ e $V = X_1 \cdots X_{n-1}$ ortogonali e Σ diagonale. Quindi il teorema è dimostrato. □

1.3 SVD troncata

La decomposizione ai valori singolari della matrice A ci permette di scrivere quest'ultima come

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

dove in A_k sono contenute le k componenti singolari principali di A.
La SVD troncata consiste nell'approssimare A con le sue prime k componenti principali, cioè

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

Quindi la matrice N può essere interpretata come rumore se i $\sigma_{k+1}, \ldots, \sigma_n$ sono molto più piccoli di σ_k.
Dal punto di vita pratico, la SVD troncata è molto importante, non solo per la rimozione del rumore, ma anche per la compressione dei dati e la risoluzione di problemi mal condizionati.
Dal punto di vista matematico, la SVD troncata ha una proprietà importante:

Teorema 1.2. *Sia $A \in \mathbb{R}^{m \times n}$ una matrice di rango $r > k$. Allora il problema di minimo*

$$\min_{\substack{Z \in \mathbb{R}^{m \times n} \\ \text{Rango}(Z) = k}} \|A - Z\|_2$$

ha come soluzione A_k, la decomposizione ai valori singolari di rango k di A

$$Z = A_k = U_k \Sigma_k V_k^T$$

Inoltre vale

$$\|A - A_k\|_2 = \sigma_{k+1}$$

Dimostrazione. Osservando che $U^T A_k V = \text{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$ segue che $\text{Rango}(A_k) = k$ e che $U^T(A - A_k)V = \text{diag}(0, \ldots, 0, \sigma_{k+1}, \ldots, \sigma_p)$ quindi si ha che $\|A - A_k\|_2 = \sigma_{k+1}$.
Sia $B \in \mathbb{R}^{m \times n}$ tale che $\text{Rango}(B) = k$. È possibile determinare una base ortonormale di vettori $x_1, \ldots, x_{n-k}$ in modo che $\ker(B) = \text{span}(x_1, \ldots, x_{n-k})$. Per ovvie ragioni

$$\text{span}(x_1, \ldots, x_{n-k}) \cap \text{span}(v_1, \ldots, v_{k+1}) \neq \{0\}$$

Sia z un vettore in questa intersezione tale che $\|z\|_2 = 1$. Allora $Bz = 0$ e $Az = \sum_{i=1}^{k+1} \sigma_i(v_i^T z)u_i$. Per la definizione *norma-2* si ha

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

Quindi, per quanto visto sopra, A_k è soluzione del problema di minimo. □

Teorema 1.3. *Sia $A \in \mathbb{R}^{m\times n}$ una matrice di rango $r > k$. Allora il problema di minimo*

$$\min_{\substack{Z\in\mathbb{R}^{m\times n} \\ \text{Rango}(Z)=k}} \|A - Z\|_F$$

ha come soluzione A_k, la decomposizione ai valori singolari di rango k di A

$$Z = A_k = U_k \Sigma_k V_k^T$$

Inoltre vale

$$\|A - A_k\|_F = \left(\sum_{k+1}^{r} \sigma_i^2 \right)^{1/2}$$

La dimostrazione di questo teorema richiede un lemma.

Lemma 1.4. *Dato il seguente prodotto scalare definito sullo spazio $\mathbb{R}^{m\times n}$*

$$\langle A, B\rangle = tr(A^T B) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} b_{ij}$$

con $A,\ B \in \mathbb{R}^{m\times n}$. Sia $A \in \mathbb{R}^{m\times n}$ con SVD $A = U\Sigma V^T$. Allora le matrici

$$u_i v_j^T, \qquad i = 1, 2, \ldots, m \qquad j = 1, 2, \ldots, n$$

costituiscono una base ortonormale di $\mathbb{R}^{m\times n}$

Dimostrazione. (Lemma 1.4) Con l'identità $\langle A, B\rangle = \text{tr}(A^T B) = \text{tr}(BA^T)$ si ha

$$\langle u_i v_j^T, u_k v_l^T\rangle = \text{tr}(v_j u_i^T u_k v_l^T) = \text{tr}(v_l^T v_j u_i^T u_k) = (v_l^T v_j)(u_i^T u_k) = 0$$

e questo mostra che le matrici sono ortonormali. Poiché tali matrici sono mn, costituiscono una base dello spazio $\mathbb{R}^{m\times n}$. □

Dimostrazione. (Teorema 1.3) La matrice $Z \in \mathbb{R}^{m\times n}$ può essere scritta in termini della base definita nel lemma precedente

$$Z = \sum_{i,j} \alpha_{ij} u_i v_j^T$$

per determinati α_{ij}. Per l'ortonormalità della base considerata si ha

$$\|A - Z\|_F^2 = \sum_{i,j} (\sigma_{ij} - \alpha_{ij})^2 = \sum_{i=j} (\sigma_{ii} - \alpha_{ii})^2 + \sum_{i\neq j} \alpha_{ij}^2$$

dove i σ_{ij} sono gli elementi della matrice Σ che contiene i valori singolari della matrice A. Ovviamente si può scegliere la matrice Z in modo che il secondo termine della sommatoria precedente sia nullo. Quindi si ha

$$Z = \sum_{i=j} \alpha_{ii} u_i v_i^T$$

Dato che il rango di Z è uguale al numero di termini di questa sommatoria, la condizione Rango$(Z) = k$ implica che ci sono esattamente k termini non nulli nella sommatoria. Per raggiungere il minimo di $\|A - Z\|_F^2$ si devono scegliere $\alpha_{ii} = \sigma_{ii}, \quad i = 1, \ldots, k$ e da ciò segue la tesi. Infatti

$$Z = \sum_{i=1}^{k} \alpha_{ii} u_i v_i^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T = A_k$$

da cui

$$\|A - Z\|_F^2 = \sum_{i=1}^{r} (\sigma_i - \alpha_i)^2 = \sum_{i=1}^{k} (\sigma_i - \sigma_i)^2 + \sum_{i=k+1}^{r} \sigma_i^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

□

I teoremi precedenti mostrano che A_k è la migliore approssimazione di rango k della matrice A. Per questo motivo la SVD troncata costituisce un termine di paragone rispetto al quale verrà valutata l'accuratezza delle decomposizioni che verranno presentate nel seguito. Tale proprietà risulta utile nella pratica poiché gli algoritmi per ricavare le prime k componenti singolari sono accurati e robusti.
Nel seguito, con $\mathcal{O}(\text{SVD}(A, k))$ verrà indicato il costo computazionale per il calcolo della SVD troncata di rango k.
Pur essendo utilizzata come riferimento per altre decomposizioni, la SVD troncata presenta alcuni svantaggi. Nelle applicazioni, soprattutto nel *text*

mining, la matrice dei dati A è sparsa e, applicando la SVD, le componenti singolari calcolate sono quasi completamente dense. Ciò significa che, memorizzare A_k, con k grande ed A sparsa, richiede più spazio che memorizzare solamente A. Inoltre, spesso la matrice A è non negativa mentre le componenti singolari calcolate dalla SVD troncata hanno anche segno negativo. Perdere le proprietà di sparsità e non negatività, che caratterizzano la matrice dei dati, rende difficile, se non impossibile, l'interpretazione dei fattori della SVD troncata.

1.4 Considerazioni sull'interpretabilità

Nonostante la SVD troncata venga ampiamente utilizzata, i vettori u_i e v_i possono non avere un significato nel campo da cui provengono i dati. Per esempio supponendo che l'autovettore

$$[(1/4)\text{età} - (1/\sqrt{2})\text{altezza} + (1/3)\text{stipendio} + (3/7)\text{peso}]$$

sia uno dei fattori principali provenienti da un dataset contenente caratteristiche di persone, non è particolarmente significativo o ricco d'informazioni. Questo accade frequentemente, poiché i vettori singolari sono astrazioni matematiche che possono essere calcolate per qualsiasi matrice. Non sono "cose" con una realtà "fisica ". Tuttavia, molto spesso, si fa un tentativo di *reificazione*, cioè si cerca di dare un significato fisico o un'interpretazione alle componenti singolari più grandi.
In alcuni casi particolari, come per esempio, in un dataset di 1000 punti del piano provenienti da una distribuzione normale multivariata, in Figura 1.1, è possibile dimostrare che le componenti principali sono le direzioni degli assi dell'ellissoide da cui sono stati presi i dati.

Tuttavia, nella maggior parte dei casi, ciò non è possibile. Per esempio,in un dataset di 1000 punti del piano provenienti dall'unione di due distribuzioni normali multivariate, in Figura 1.2, non è possibile interpretare le componenti principali come è stato fatto nel caso precedente. In questo ed altri esempi potrebbe essere difficile interpretare questi vettori in modo significativo in termini dei processi che hanno generato i dati.

Nonostante in certi casi la *reificazione* sia giustificata, la necessità di un'interpretazione non può nascere solamente dalla matematica, ma richiede una conoscenza profonda del campo da cui provengono i dati.

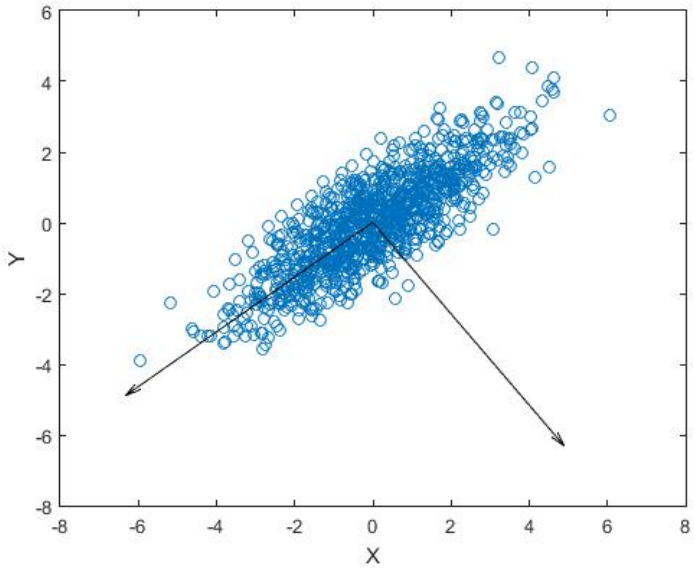

Figura 1.1: Scatter plot di dati provenienti una distribuzione normale multivariata

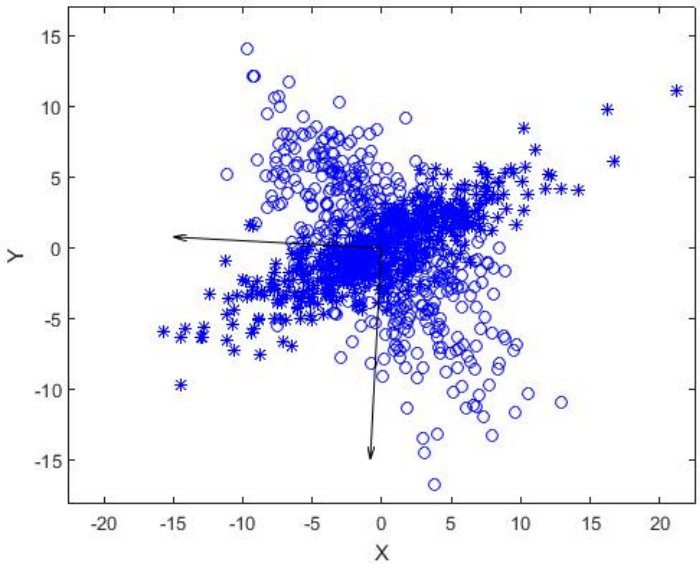

Figura 1.2: Scatter plot di dati provenienti dall'unione di due distribuzioni normali multivariate

1.5 Problema dei minimi quadrati

Il problema dei minimi quadrati ha applicazioni in molti ambiti del calcolo scientifico. In particolare, tale problema,viene utilizzato per approssimare la soluzione di un sistema $Ax = b$ sovradeterminato, cioè con un numero di equazioni superiore al numero di incognite.
Data $A \in \mathbb{R}^{m\times n}$, con $m \geq n$ e $b \in \mathbb{R}^m$, il problema dei minimi quadrati consiste nel determinare il vettore $x \in \mathbb{R}^n$ soluzione del problema di ottimizzazione

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 \tag{1.1}$$

Mostriamo ora un metodo risolutivo, valido nel caso in cui A abbia rango massimo, che trasforma il problema di minimo in un sistema lineare detto *sistema normale*

Teorema 1.5. *Data $A \in \mathbb{R}^{m\times n}$, con $m \geq n$ e $b \in \mathbb{R}^m$ diciamo che $\bar{x} \in \mathbb{R}^n$ è soluzione di (1.1) se e solo se è soluzione del sistema normale*

$$A^T Ax = A^T b \tag{1.2}$$

Dimostrazione. $\bar{x} \in \mathbb{R}^n$ è soluzione di (1.1) se

$$\Phi(\bar{x}) = \|A\bar{x} - b\|_2^2 \leq \|Ax - b\|_2^2 = \Phi(x) \qquad \forall x \in \mathbb{R}^n$$

La soluzione di tale problema può essere ottenuta imponendo che il gradiente della funzione Φ si annulli in $\bar{x}$. Essendo

$$\Phi(x) = (Ax - b)^T(Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b$$

si trova allora

$$\bigtriangledown\Phi(\bar{x}) = 2A^T A\bar{x} - 2A^T b = 0$$

da cui si deduce che $\bar{x}$ deve essere soluzione di (1.2). Tale sistema è non singolare se A ha rango massimo. In tal caso la soluzione esiste ed è unica. □

Se A non ha rango massimo la risoluzione mediante (1.2) risulta inappropriata.

Osservazione. Se A non ha rango massimo la soluzione non è unica, infatti supponendo che $\bar{x}$ sia soluzione di (1.1) anche $\bar{x}+z$ con $z \in \ker(A)$ è soluzione.

Per questo motivo è necessario introdurre un ulteriore vincolo per determinare un'unica soluzione. Perciò si richiede che la soluzione $\bar{x}$ abbia norma euclidea minima ed il problema diventa

$$\begin{aligned} &\text{trovare } \bar{x} \in \mathbb{R}^n \text{ tale che } \|\bar{x}\|_2 \text{ sia minima e} \\ &\|A\bar{x} - b\|_2^2 \leq \|Ax - b\|_2^2 \qquad \forall x \in \mathbb{R}^n \end{aligned} \tag{1.3}$$

Diamo ora la definizione di matrice pseudo-inversa mostriamo il suo utilizzo nella risoluzione del problema (1.1) nel caso generale.

Definizione 1.6. Sia $A \in \mathbb{R}^{m\times n}$, la *pseudo-inversa* di A, indicata con A^+, è l'unica matrice $n \times m$ che verifica le seguenti proprietà:

- $AA^+A = A$
- $A^+AA^+ = A^+$
- $(AA^+)^T = AA^+$
- $(A^+A)^T = A^+A$

Osservazione. Se $A \in \mathbb{R}^{m\times m}$ ed è invertibile allora $A^+ = A^{-1}$

Se la matrice $A \in \mathbb{R}^{m\times n}$ ha rango massimo esiste una formula esplicita per determinare la pseudo-inversa

- se $m \geq n$ allora $A^+ = (A^TA)^{-1}A^T$ ed è un'inversa sinistra cioè $A^+A = I$
- se $m \leq n$ allora $A^+ = A^T(AA^T)^{-1}$ ed è un'inversa destra cioè $AA^+ = I$

In generale se $A \in \mathbb{R}^{m\times n}$ tale che $\text{Rango}(A) = r \leq \min\{m, n\}$, utilizzando la SVD si ha

$$A = U\Sigma V^T = \begin{pmatrix} U_r & U_r^\perp \end{pmatrix} \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_r^T \\ V_r^{\perp T} \end{pmatrix} = U_r\Sigma_r V_r^T$$

dove $\Sigma_r \in \mathbb{R}^{r\times r}$ è una matrice diagonale con tutti gli elementi diagonali non nulli, quindi Σ_r è invertibile. Da tale scrittura segue che la pseudo-inversa della matrice iniziale è

$$A^+ = V_r\Sigma_r^{-1}U_r^T$$

Teorema 1.6. *Sia $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$ allora l'unica soluzione di (1.3) è $\bar{x} = A^+b$.*

Dimostrazione. Utilizzando la SVD di A, $A = U\Sigma V^T$, il problema (1.3) è equivalente a cercare $w = V^Tx$ tale che w abbia norma euclidea minima e

$$\|\Sigma w - U^Tb\|_2^2 \leq \|\Sigma y - U^Tb\|_2^2 \qquad \forall y \in \mathbb{R}^n$$

Se r è il numero di valori singolari σ_i non nulli di A, allora

$$\|\Sigma w - U^Tb\|_2^2 = \sum_{i=1}^{r}(\sigma_i w_i - (U^Tb)_i)^2 + \sum_{i=r+1}^{m}((U^Tb)_i)^2$$

che è minimo se $w_i = (U^Tb)_i\sigma_i$ per $i = 1, \ldots, r$. Inoltre, fra tutti i vettori $w \in \mathbb{R}^n$ che hanno le prime r componenti fissate, quello che rende minima la norma euclidea ha le restanti $n - r$ componenti nulle. Quindi, il vettore soluzione cercato è $\bar{w} = \Sigma^+U^Tb$ ossia $\bar{x} = V\Sigma^+U^Tb = A^+b$. □

Introduciamo ora il problema dei minimi quadrati multidimensionale, strettamente legato a (1.3), utilizzato frequentemente nella regressione lineare multipla. Data $A \in \mathbb{R}^{m\times n}$ con $m \geq n$ e $B \in \mathbb{R}^{m\times p}$, il problema dei minimi quadrati multidimensionale consiste nel determinare la matrice $\widetilde{X} \in \mathbb{R}^{n\times p}$ soluzione del problema di ottimizzazione

$$\min_{X\in\mathbb{R}^{n\times p}} \|B - AX\|_F \tag{1.4}$$

Teorema 1.7. *Data $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{m\times p}$ allora l'unica soluzione di (1.4) è $\widetilde{X} = A^+B$.*

Dimostrazione. Utilizzando il Teorema 1.6 si va a risolvere

$$\min_{x_j\in\mathbb{R}^n} \|B^{(j)} - Ax_j\|_2$$

per ogni colonna $B^{(j)}$, $\quad j = 1, \ldots, p$ si ottengono le colonne di $\widetilde{X}$ e si ha la tesi. □

Tale teorema afferma che $\widetilde{X} = A^+B$ è la più "piccola" matrice che risolve il problema di ottimizzazione, utilizzando tutte le informazioni contenute nella matrice B.

Capitolo 2

Decomposizione CUR

La decomposizione CUR è una fattorizzazione che permette di scrivere una data matrice $A \in \mathbb{R}^{m \times n}$ come prodotto di tre matrici C, U e R dove C ed R sono formate rispettivamente da colonne e righe di A, mentre U è costruita appositamente per garantire che il prodotto CUR sia "vicino" ad A. Perciò, dato che le colonne di A sono contenute in C ed allo stesso modo le righe di A sono contenute in R, possono essere usate al posto dei vettori singolari destri e di quelli sinistri, con il beneficio di una migliore interpretabilità in termini dei dati di partenza. Inoltre, nel seguito verrà dimostrato che la decomposizione CUR sarà buona quasi quanto la migliore decomposizione di rango basso di A ottenuta mediante la SVD troncata.
Sono molteplici i motivi per i quali è preferibile un'approssimazione di rango basso espressa in termini di un numero esiguo di colonne e/o righe della matrice di partenza:

- Proprietà della matrice, come la sparsità e la non negatività, vengono mantenute.
- In alcune applicazioni tutti dati vengono espressi in termini di determinati elementi del dataset ritenuti significativi.
- Per matrici estremamente grandi, per le quali il calcolo della SVD è troppo oneroso, si preferisce una decomposizione come quella descritta sopra.
- Risulta più facile analizzare ed interpretare i risultati della decomposizione CUR.
- La matrice di partenza viene "compressa" occupando così meno spazio in memoria.

Seguono ora alcune definizioni che verranno utilizzate nel seguito.

Definizione 2.1. Data $A \in \mathbb{R}^{m\times n}$, per ogni matrice $C \in \mathbb{R}^{m\times c}$, le cui colonne sono c colonne della matrice A, la matrice $A' \in \mathbb{R}^{m\times n}$, $A' = CX$ è detta *matrice di approssimazione per colonne ad* A, per qualsiasi matrice $X \in \mathbb{R}^{c\times n}$.
Bisogna notare alcune cose in questa definizione:

- Non viene data nessuna informazione in merito alla struttura della matrice X.
- Si è interessati ai casi in cui $c \ll n$ e a seconda del contesto, c sarà costante ed indipendente da n o dipendente logaritmicamente da n.
- Ogni colonna di A' è una combinazione lineare delle "colonne base" che sono le colonne di A.

La seguente proposizione da informazioni sulla struttura della matrice X.

Proposizione 2.1. *Dato un insieme di colonne C, la matrice di approssimazione $A' = P_C A = CC^+A$ (dove $P_C A = CC^+$ è la proiezione di A nel sottospazio generato dalle colonne di C) soddisfa la definizione 2.1 ed in particolare, per quanto detto nel Teorema 1.7 è la "migliore" approssimazione per colonne ad A, cioè $X_{opt} = C^+A$ è soluzione del problema di minimo*

$$\min_{X\in\mathbb{R}^{c\times n}} \|A - CX\|_F \tag{2.1}$$

Definizione 2.2. Data $A \in \mathbb{R}^{m\times n}$, per ogni matrice $C \in \mathbb{R}^{m\times c}$, le cui colonne sono c colonne della matrice A, sia $X = C^+A$, si definisce la *decomposizione* CX di A come la matrice di approssimazione per colonne $A' = CX$.

Definizione 2.3. Sia $A \in \mathbb{R}^{m\times n}$, per ogni matrice $C \in \mathbb{R}^{m\times c}$ le cui colonne sono c colonne della matrice A, ed ogni matrice $R \in \mathbb{R}^{r\times n}$ le cui righe sono r righe della matrice A, la matrice $A' \in \mathbb{R}^{m\times n}$, $A' = CUR$ è detta *matrice di approssimazione per colonne-righe ad* A, per qualsiasi matrice $U \in \mathbb{R}^{c\times r}$.
Bisogna notare alcune cose in questa definizione:

- Non viene data nessuna informazione in merito alla struttura della matrice U.
- Una matrice di approssimazione per righe-colonne ad A è una particolare matrice di approssimazione per righe, per la quale la matrice X viene espressa come combinazione lineare di alcune righe di A.

- Lo spazio in memoria occupato dalla combinazione di C, U ed R è minore di quello occupato da A' quando c, r sono molto più piccoli di n, m.

La seguente proposizione da informazioni sulla struttura della matrice U.

Proposizione 2.2. *Dato un insieme di colonne C ed un insieme di righe R di A la matrice di approssimazione $A' = C(C^+AR^+)R$ soddisfa la definizione 2.3 ed in particolare è la "migliore" approssimazione per colonne-righe ad A, cioè la matrice $\tilde{U} = C^+AR^+$ è soluzione del problema di minimo*

$$\min_{U \in \mathbb{R}^{c \times r}} \|A - CUR\|_F \tag{2.2}$$

Dimostrazione. Dalla Proposizione 2.1 è noto che $X_{opt} = C^+A$ è soluzione del problema (2.1). Da qui si cerca un'approssimazione per righe di X_{opt} risolvendo il problema di minimo

$$\min_{U \in \mathbb{R}^{c \times r}} \|X_{opt} - UR\|_F = \min_{U \in \mathbb{R}^{c \times r}} \|X_{opt}^T - R^T U^T\|_F \tag{2.3}$$

Per il Teorema 1.7, la soluzione di tale problema è

$$U_{opt}^T = (R^T)^+ X_{opt}^T = (R^+)^T X_{opt}^T = (X_{opt} R^+)^T = (C^+AR^+)^T$$

da cui segue che $U_{opt} = C^+AR^+$. □

Definizione 2.4. Data $A \in \mathbb{R}^{m \times n}$, per ogni matrice $C \in \mathbb{R}^{m \times c}$ le cui colonne sono c colonne della matrice A, ed ogni matrice $R \in \mathbb{R}^{r \times n}$ le cui righe sono r righe della matrice A, sia $U = C^+AR^+$, si definisce la *decomposizione CUR* di A come la matrice di approssimazione per colonne-righe $A' = CUR$.

Si può osservare che la struttura della decomposizione CUR è simile a quella della SVD troncata, eccetto che per le proprietà delle matrici C ed R.

$$\underbrace{\begin{pmatrix} \\ A \\ \\ \end{pmatrix}}_{m \times n} \approx \underbrace{\begin{pmatrix} \\ C \\ \\ \end{pmatrix}}_{m \times c} \underbrace{\begin{pmatrix} \\ U \\ \\ \end{pmatrix}}_{c \times r} \underbrace{\begin{pmatrix} \\ R \\ \\ \end{pmatrix}}_{r \times n}$$

2.1 Ricerche e lavori precedenti

Nei paragrafi seguenti verranno brevemente presentati alcuni lavori intermedi nei campi dell'algebra lineare numerica e dell'informatica teorica che hanno portato alla decomposizione CUR.

2.1.1 Ricerche di algebra lineare numerica

Molti esperti di algebra lineare numerica hanno studiato decomposizioni matriciali con struttura e proprietà simili alle decomposizioni CX e CUR che sono state definite in precedenza. La maggior parte dei lavori è legata alla decomposizione QR.
Stewart ed i suoi collaboratori erano interessati a calcolare un'approssimazione sparsa di basso rango di grandi matrici termini-documento [4, 5, 6]. Sviluppò il metodo *quasi-Gram-Schmidt* come variante della decomposizione QR: data in input una matrice $A \in \mathbb{R}^{m\times n}$ e un parametro k, che indica il rango, viene restituita una matrice $C \in \mathbb{R}^{m\times k}$ costituita da k colonne di A che costituiscono una base per l'approssimazione dello spazio delle colonne di A ed una matrice $T_C \in \mathbb{R}^{k\times k}$ che rende ortogonali tali colonne. La matrice $Q_C = CT_C^{-1}$ non viene calcolata esplicitamente. Tale metodo fornisce una decomposizione della forma $A \approx CX$. Applicando il metodo ad A per ottenere C ed ad A^T per ottenere $R \in \mathbb{R}^{k\times n}$, costituita da k righe di A, è possibile determinare la matrice U in modo che $A \approx CUR$ minimizzando $\|A - CUR\|_F^2$. Nonostante tale approssimazione non abbia garanzie teoriche, l'analisi dell'errore all'indietro mostra che il metodo si comporta bene.

Goreinov, Tyrtyshnikov e Zamarashkin erano interessati ad applicazioni come lo scattering, dove matrici molto grandi hanno blocchi che possono essere facilmente approssimati da matrici di rango basso. Mostrarono che se la matrice A è approssimata da una matrice di rango k con un'accuratezza ε allora esiste una scelta di k colonne e k righe, cioè esistono C ed R, da cui costruire $U \in \mathbb{R}^{k\times k}$ in modo che $A \approx CUR$ con un'approssimazione $\|A - CUR\|_2 \leq \varepsilon f(m,n,k)$ dove $f(m,n,k) = 1 + 2\sqrt{km} + 2\sqrt{kn}$. La scelta di queste matrici viene fatta seguendo diverse strategie, ognuna delle quali va a minimizzare o massimizzare una certa quantità [7, 8, 9] come il volume del parallelepipedo che ha per lati i vettori scelti.

Gu ed Eisenstat, nel loro articolo più influente [10], descrivono una fattorizzazione QR strong rank-revealing che seleziona in modo deterministico esattamente k colonne da una matrice $A \in \mathbb{R}^{m\times n}$. Gli algoritmi di questo tipo sono efficienti, infatti il loro costo computazionale è $\mathcal{O}(mn^2)$ (assumendo che $m \geq n$), che è lo stesso costo della SVD di A. Inoltre, Gu ed Eisenstat hanno

dimostrato che se la matrice $C \in \mathbb{R}^{m\times k}$ contiene le k colonne scelte (senza scaling), allora $\sigma_{\min}(C) \geq \sigma_k(A)/f(k,n)$, dove $f(k,n) = \mathcal{O}(\sqrt{k(n-k)})$.

Infine, recentemente, Martinsson, Rokhlin e Tygert [11] hanno proposto un metodo per calcolare efficientemente un'approssimazione vicina alla miglior approssimazione di rango k (cioè alla SVD troncata di rango k) di una matrice $A \in \mathbb{R}^{m\times n}$. Il cuore del loro algoritmo è un metodo di proiezione casuale, che proietta A su uno spazio generato da un piccolo numero (che chiameremo l) di vettori casuali; i vettori vengono generati casualmente a partire da una distribuzione Gaussiana di media zero e varianza unitaria. La formula generale del loro risultato è piuttosto complicata, ma ponendo $l = k + 20$, è possibile costruire un'approssimazione A' di rango k tale che

$$\|A - A'\|_2 \leq 10\sqrt{(k+20)m}\|A - A_k\|_2$$

è verificata con una probabilità almeno $1-10^{-17}$. Inoltre, gli autori estendono il loro algoritmo per calcolare la decomposizione interpolativa ID di una matrice A. Questa decomposizione viene espressa esplicitamente in termini di un esiguo numero di colonne di A, ed è una versione più restrittiva della decomposizione CX. Più precisamente, si richiede che ogni elemento della matrice X sia in valore assoluto minore di una piccola costante (ad esempio due).

2.1.2 Ricerche di informatica teorica

Nel campo dell'informatica, molte ricerche hanno seguito l'influente lavoro di Frieze, Kannan e Vempala [12]. Il loro metodo consiste nel campionare le colonne di A per formare la matrice C in modo che

$$\|A - CX\|_\xi \leq \|A - A_k\|_\xi + \varepsilon\|A\|_F$$

per $\xi = 2, F$, fornendo così garanzie sull'errore per un'approssimazione per colonne di basso rango.

Drineas, Kannan e Mahoney hanno analizzato l'errore di approssimazione per alcuni dei loro algoritmi di decomposizione CUR. In particolare, in [13], viene calcolata l'approssimazione di una matrice $A \in \mathbb{R}^{m\times n}$ campionando c colonne ed r righe da A per formare le matrici $C \in \mathbb{R}^{m\times c}$ ed $R \in \mathbb{R}^{r\times n}$. A partire da C ed R viene costruita una matrice $U \in \mathbb{R}^{c\times r}$ con opportune proprietà in modo che

$$\|A - CUR\|_\xi \leq \|A - A_k\|_\xi + \varepsilon\|A\|_F$$

sia verificata con alta probabilità per $\xi = 2, F$.

Dipendentemente dalle proprietà della matrice di partenza e dal modo in cui vengono scelte le colonne, si può modificare l'algoritmo migliorando l'errore delle decomposizione CX e CUR.

I lavori di ricerca più recenti hanno portato ad abbassare il costo computazionale degli algoritmi e la loro complessità.

2.2 Subspace sampling

Per quanto detto nelle Proposizioni 2.1 e 2.2, le matrici X ed U sono definite a partire dalle matrici C ed R. Quindi la parte centrale degli algoritmi di decomposizione, è la scelta delle colonne-righe della matrice di partenza A. La tecnica di sampling che verrà descritta è comunemente chiamata *subspace sampling*. In particolare verranno descritti gli algoritmi COLUMNSELECT e ROWSELECT che poi saranno utilizzati per costruire l'algoritmo ALGORITHMCUR.

Nei primi articoli sull'argomento [13], le colonne di A venivano estratte secondo una distribuzione di probabilità che dipendeva dalla norma euclidea delle colonne stesse. Invece, con il *subspace sampling* si estraggono in modo random le colonne di A secondo una distribuzione di probabilità che dipende dalla norma euclidea delle righe della matrice dei primi k vettori singolari destri di A. Il primo algoritmo presentato si chiama COLUMNSELECT e prende in input una matrice $A \in \mathbb{R}^{m\times n}$, un parametro di rango k ed un parametro d'errore ε e restituisce come output una matrice $C \in \mathbb{R}^{m\times c}$ costituita da alcune colonne di A scalate.

Definizione 2.5. Data $A \in \mathbb{R}^{m\times n}$, sia $V_k \in \mathbb{R}^{n\times k}$ la matrice costituita dai primi k vettori singolari destri della matrice A ottenuti mediante la SVD troncata, allora si definiscono *probabilità di campionamento per le colonne* di A gli scalari

$$p_i = \frac{1}{k}\|(V_k)_{(i)}\|_2^2 \qquad \forall i = 1, \dots, n \tag{2.4}$$

dove $(V_k)_{(i)}$ indica l'i-esima riga della matrice V_k.

Osservazione 1. Per la definizione delle *probabilità di campionamento per le colonne* in (2.4) si ha che $\sum_{i=1}^{n} p_i = 1$. Infatti, essendo le colonne di V_k ortonormali, si ha

$$\sum_{i=1}^{n} p_i = \frac{1}{k}\sum_{i=1}^{n}\|(V_k)_{(i)}\|_2^2 = \frac{1}{k}\sum_{j=1}^{k}\|(V_k)^{(j)}\|_2^2 = 1$$

dove $(V_k)^{(j)}$ indica la j-esima colonna della matrice V_k.

L'algoritmo COLUMNSELECT è molto semplice in quanto calcola le probabilità di campionamento per le righe e chiama uno dei due algoritmi ausiliari di sampling presentati a seguire. Tali algoritmi sono strettamente collegati e differiscono solamente per il modo in cui vengono scelte le colonne.
L'algoritmo EXACTLY(c) sceglie esattamente c colonne di A per formare C facendo c estrazioni indipendenti identicamente distribuite (i.i.d), dove per ogni estrazione la probabilità che la colonna i-esima di A venga presa è p_i.
Invece, l'algoritmo EXPECTED(c) sceglie in media c colonne di A per formare C, la probabilità che la colonna i-esima di A venga inserita in C è $\bar{p_i} = \min\{1, cp_i\}$.
Un'ulteriore differenza tra EXACTLY(c) e EXPECTED(c) è che con il primo algoritmo la stessa colonna può essere campionata più di una volta, mentre con il secondo le colonne vengono campionate una sola volta. Per poter effettuare delle manipolazioni algebriche la matrice C verrà scritta come il prodotto di tre matrici $C = AS_CD_C$. La matrice S_C viene chiamata *matrice di sampling* mentre D_C *matrice di scaling*. Per ulteriori dettagli riguardo i due algoritmi di sampling e le proprietà delle matrici S_C e D_C si veda in Appendice A. Il costo computazionale dell'algoritmo COLUMNSELECT è dominato dal calcolo delle probabilità di campionamento (2.4), il cui costo è $\mathcal{O}(\text{SVD}(A, k))$. I primi k vettori singolari destri di A possono essere calcolati in modo efficiente (ed approssimato) utilizzando funzioni già implementate in Matlab e nel seguito, il calcolo della SVD verrà trattato come una "black box".

Input $A \in \mathbb{R}^{m\times n}$, un parametro di rango k, un parametro d'errore ε.

Output $C \in \mathbb{R}^{m\times c}$.

- Calcola le probabilità di campionamento p_i per $i = 1, \ldots, n$ definite in (2.4)
- (Implicitamente) costruisce una matrice di sampling S_C ed una matrice diagonale di scaling D_C usando l'algoritmo EXACTLY(c) o l'algoritmo EXPECTED(c)
- Restituisce $C = AS_CD_C$

Algoritmo COLUMNSELECT

Osservazione 2. Per campionare le righe di A si potrebbe utilizzare COLUMN-SELECT su A^T e poi prendere la trasposta della matrice ottenuta in output. Da tale osservazione segue che

$$R = (A^T S_R D_R)^T = D_R^T S_R^T A$$

e se D_R è una matrice diagonale allora $R = D_R S_R^T A$.

Dall'osservazione precedente si ha la seguente proposizione che da informazioni sulla struttura della matrice U.

Proposizione 2.3. *Data $A \in \mathbb{R}^{m\times n}$, $C = AS_C D_C$ matrice formata dalle colonne di A e $R = D_R^T S_R^T A$ matrice formata dalle righe di A allora $U = (D_R^T S_R^T C)^+$ è soluzione del problema di minimo*

$$\min_{U\in\mathbb{R}^{c\times r}} \|A - CUR\|_F$$

Dimostrazione. Per la Proposizione 2.2 la soluzione del problema di minimo è la matrice $\tilde{U} = C^+ A R^+$. Sostituendo C ed R e per le proprietà della pseudo-inversa si ha che

$$\begin{aligned}\tilde{U} &= (S_C D_C)^+ A^+ A A^+ (D_R^T S_R^T)^+ = (S_C D_C)^+ A^+ (D_R^T S_R^T)^+ \\ &= (AS_C D_C)^+ (D_R^T S_R^T)^+ = C^+ (D_R^T S_R^T)^+ = (D_R^T S_R^T C)^+\end{aligned}$$

□

Banalmente si può notare che la matrice dei primi k vettori singolari destri di A^T coincide con la matrice dei primi k vettori singolari sinistri di A, cioè se $A \approx U_k \Sigma_k V_k^T$ allora $A^T \approx V_k \Sigma_k U_k^T$. Per tale motivazione risulta utile dare la seguente definizione.

Definizione 2.6. Data $A \in \mathbb{R}^{m\times n}$, sia $U_k \in \mathbb{R}^{m\times k}$ la matrice costituita dai primi k vettori singolari sinistri della matrice A ottenuti mediante la SVD troncata, allora si definiscono *probabilità di campionamento per le righe* di A gli scalari

$$p_i = \frac{1}{k}\|(U_k)_{(i)}\|_2^2 \qquad \forall i = 1, \ldots, m \tag{2.5}$$

dove $(U_k)_{(i)}$ indica l'i-esima riga della matrice U_k.

Osservazione 3. Per la definizione delle *probabilità di campionamento per le righe* in (2.5) si ha che $\sum_{i=1}^m p_i = 1$. Infatti, essendo le colonne di U_k ortonormali,

$$\sum_{i=1}^m p_i = \frac{1}{k}\sum_{i=1}^m \|(U_k)_{(i)}\|_2^2 = \frac{1}{k}\sum_{j=1}^k \|(U_k)^{(j)}\|_2^2 = 1$$

dove $(U_k)^{(j)}$ indica la j-esima colonna della matrice U_k.

Sfruttando le osservazioni appena fatte si può costruire l'algoritmo ROWSELECT che prende in input una matrice $A \in \mathbb{R}^{m\times n}$, una matrice $C \in \mathbb{R}^{m\times c}$ formata da colonne scalate di A, un parametro d'errore ε, un intero r e restituisce come output una matrice $R \in \mathbb{R}^{r\times n}$ costituita da alcune righe di A, una matrice $W \in \mathbb{R}^{r\times c}$ formata dalle righe corrispondenti di C e la pseudo-inversa di U. In modo analogo al COLUMNSELECT, l'algoritmo prima calcola le probabilità di campionamento per le righe e chiama uno dei due algoritmi ausiliari di sampling per estrarre alcune righe di A, poi costruisce la matrice U come detto nella Proposizione 2.3.

Input $A \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{m\times c}$ formata da colonne di A, un parametro d'errore ε ed un numero intero $r \leq m$

Output $R \in \mathbb{R}^{r\times n}$ formata da r righe di A, $W \in \mathbb{R}^{c\times r}$ formata dalle corrispondenti r righe di C ed $U \in \mathbb{R}^{r\times c}$

- Calcola le probabilità di campionamento p_i per $i = 1, \ldots, m$ definite in (2.5)
- (Implicitamente) costruisce una matrice di sampling S_R ed una matrice diagonale di scaling D_R usando l'algoritmo EXACTLY(c) o l'algoritmo EXPECTED(c)
- Costruisce e restituisce la matrice $R = D_R^T S_R^T A$ formata da alcune colonne riscalate di A
- Costruisce e restituisce la matrice $W = D_R^T S_R^T C$ formata da alcune colonne riscalate di C
- Restituisce $U = W^+$

Algoritmo ROWSELECT

L'algoritmo ALGORITHMCUR è particolarmente semplice poiché si limita a richiamare prima il COLUMNSELECT e poi il ROWSELECT con i parametri dati in input. Quindi ALGORITHMCUR prende in input una matrice $A \in \mathbb{R}^{m\times n}$, un parametro di rango k, un parametro d'errore $\varepsilon \in (0, 1]$ ed un intero $r \leq m$ e restituisce la terna C, U, R, dove C è formata da alcune colonne di A mentre R da alcune righe.
Per le implementazioni in Matlab degli algoritmi descritti si veda in Appendice B.

2.3 Analisi dell'errore

Prima di enunciare e dimostrare alcuni teoremi riguardanti l'errore delle decomposizioni CX e la CUR, risulta utile proporre un algoritmo per la risoluzione approssimata del problema dei minimi quadrati multidimensionale (1.4). Tale algoritmo verrà chiamato SOLVEL2 e prende in input una matrice $A \in \mathbb{R}^{m\times n}$ di rango minore uguale di k, una matrice $B \in \mathbb{R}^{m\times p}$ ed un intero $r \leq m$ restituendo una matrice $\tilde{X}_{opt} \in \mathbb{R}^{n\times p}$ ed un numero $\tilde{\mathcal{Z}} \in \mathbb{R}$.

Input $A \in \mathbb{R}^{m\times n}$ con Rango$(A) \leq k$ e $p_i \geq 0$ per $i = 1, \ldots, n$ tali che $\sum_{i=1}^{n} p_i = 1$, $B \in \mathbb{R}^{m\times p}$ ed un intero $r \leq m$.

Output Una matrice $\tilde{X}_{opt} \in \mathbb{R}^{n\times p}$, $\tilde{\mathcal{Z}} \in \mathbb{R}$.

- Calcola le probabilità di campionamento p_i per $i = 1, \ldots, m$ definite in (2.5)
- (Implicitamente) costruisce una matrice di sampling S ed una matrice diagonale di scaling D usando EXACTLY(c) o EXPECTED(c)
- Costruisce la matrice DS^TA formata da alcune righe scalate di A
- Costruisce la matrice DS^TB formata da alcune righe scalate di B
- Restituisce $\tilde{X}_{opt} = (DS^TA)^+DS^TB$
- Restituisce $\tilde{\mathcal{Z}} = \min_{X\in\mathbb{R}^{n\times p}} \|DS^TB - DS^TA\tilde{X}_{opt}\|_F$

Algoritmo SOLVEL2

Presentiamo ora un teorema sull'errore dell'algoritmo SOLVEL2. Per la dimostrazione si veda [14].

Teorema 2.4. *Sia $A \in \mathbb{R}^{m\times n}$ una matrice di rango al massimo k, $B \in \mathbb{R}^{m\times p}$ e $\mathcal{Z} = \min_{X\in\mathbb{R}^{n\times p}} \|B - AX\|_F = \|B - AX_{opt}\|_F$ dove $X_{opt} = A^+B = A_k^+B$ per le ipotesi sul rango di A. Eseguendo l'algoritmo* SOLVEL2*, se vengono scelte esattamente $r = 3200k^2/\varepsilon^2$ righe mediante l'algoritmo* EXACTLY*(c) allora almeno con probabilità* 0.7 *si ha che*

$$\|B - A\tilde{X}_{opt}\|_F \leq (1+\varepsilon)\mathcal{Z} \tag{2.6}$$

In modo simile, con le stesse ipotesi, se vengono scelte in media $r = O(k \log k/\varepsilon^2)$ *righe mediante l'algoritmo* EXACTLY*(c) allora almeno con probabilità* 0.7 *vale (2.6).*

Tale teorema verrà utilizzato nella dimostrazione dei teoremi seguenti. Presentiamo ora un teorema sull'errore della decomposizione CX ottenuta mediante l'algoritmo COLUMNSELECT.

Teorema 2.5. *Eseguendo* COLUMNSELECT *con input una matrice* $A \in \mathbb{R}^{m\times n}$*, un intero* $k \ll \min\{m, n\}$ *e* $\varepsilon \in (0, 1]$ *in input e ponendo* $c = 3200k^2/\varepsilon^2$*, se vengono scelte esattamente* c *(o in media* $c = \mathcal{O}(k \log k/\varepsilon^2)$*) colonne di* A *per costruire* C *allora almeno con probabilità* 0.7 *si ha che*

$$\|A - CC^+A\|_F \leq (1 + \varepsilon)\|A - A_k\|_F \tag{2.7}$$

Dimostrazione. Per ogni insieme di colonne $C = AS_CD_C$, la soluzione del problema di minimo $\min_{X\in\mathbb{R}^{c\times n}}\|A - CX\|_F$ è $X_{opt} = C^+A$. Indicando con $P_{A_k} = U_kU_k^T$ la matrice di proiezione sullo spazio generato dai primi k vettori singolari sinistri di A si ha che $P_{A_k}A = A_k$. Quindi segue che

$$\begin{aligned}\|A - CC^+A\|_F &= \|A - (AS_CD_C)(AS_CD_C)^+A\|_F \\ &\leq \|A - (AS_CD_C)(P_{A_k}AS_CD_C)^+P_{A_k}A\|_F \end{aligned}\tag{2.8}$$

Per maggiorare (2.8) è possibile utilizzare il Teorema 2.4 per determinare una soluzione approssimata del problema $\min_{X\in\mathbb{R}^{m\times n}}\|A - XA_k\|_F$ scegliendo casualmente colonne di A_k e di A in quanto $\text{Rango}(A_k) \leq k$, si ha

$$\|A-(AS_CD_C)(A_kS_CD_C)^+A_k\|_F \leq (1+\varepsilon)\|A-AA_k^+A_k\|_F = (1+\varepsilon)\|A-A_k\|_F$$

□

Per semplificare la trattazione, l'algoritmo COLUMNSELECT ed il Teorema 2.5 sono stati presentati in una versione particolare in cui (2.7) vale con probabilità costante. Il seguente teorema afferma invece che l'algoritmo può essere migliorato in modo che (2.7) valga con probabilità $1 - \delta$.

Teorema 2.6. *Dati una matrice* $A \in \mathbb{R}^{m\times n}$*,un intero* $k \ll \min\{m, n\}$*,* $\varepsilon \in (0, 1]$ *e* $\delta \in (0, 1]$*, allora esistono algoritmi randomizzati tali che o vengono scelte esattamente* $c = \mathcal{O}(k^2 \log(1/\delta)/\varepsilon^2)$ *colonne di* A *per costruire* C*, o vengono scelte in media* $c = \mathcal{O}(k \log k \log(1/\delta)/\varepsilon^2)$ *colonne per costruire* C*, per i quali valga almeno con probabilità* $1 - \delta$

$$\min_{X\in\mathbb{R}^{c\times n}}\|A - CX\|_F = \|A - CC^+A\|_F \leq (1 + \varepsilon)\|A - A_k\|_F$$

Dimostrazione. Il Teorema 2.5 afferma che la probabilità che per ogni esecuzione non valga (2.7) è minore di $0.3 \leq 1/e$ (dove e è il numero di Nepero). Eseguendo $\ln(1/\delta)$ volte l'algoritmo COLUMNSELECT, in modo che ogni esecuzione sia indipendente dall'altra, e prendendo la matrice C in modo che $\|A - CC^+A\|_F$ sia il più piccolo tra quelli ottenuti nelle varie esecuzioni, allora la probabilità che l'algoritmo sbagli tutte le volte è minore di $(1/e)^{\ln(1/\delta)} = \delta$. Quindi

$$\|A - CC^+A\|_F \leq (1+\varepsilon)\|A - A_k\|_F$$

vale almeno con probabilità $1 - \delta$. L'algoritmo descritto verifica l'enunciato del teorema e la sua implementazione in Matlab è descritta in Appendice B.4. □

Presentiamo ora un teorema sull'errore della decomposizione CUR ottenuta mediante l'algoritmo ROWSELECT.

Teorema 2.7. *Eseguendo* ROWSELECT *con input una matrice $A \in \mathbb{R}^{m\times n}$, una matrice $C \in \mathbb{R}^{m\times c}$ formata da c colonne di A e $\varepsilon \in (0,1]$ in input e ponendo $r = 3200c^2/\varepsilon^2$, se vengono scelte esattamente r (o in media $r = \mathcal{O}(c\log c/\varepsilon^2)$) righe di A e rispettivamente di C allora almeno con probabilità* 0.7 *vale*

$$\|A - CUR\|_F \leq (1+\varepsilon)\|A - CC^+A\|_F \tag{2.9}$$

Dimostrazione. Si ricorre al Teorema 2.4 per approssimare la soluzione approssimata del problema $\min_{X\in\mathbb{R}^{c\times n}}\|A - CX\|_F$ scegliendo in modo casuale le righe di A e di C. Segue da (2.6) che

$$\|A - C(D_RS_R^TC)^+D_RS_R^TA\|_F \leq (1+\varepsilon)\|A - CC^+A\|_F$$

dove $R = D_RS_R^TA$ ed $U = (D_RS_R^TC)^+$, quindi si ha (2.9). □

Per semplificare la trattazione, l'algoritmo ROWSELECT ed il Teorema 2.7 sono stati presentati in una versione particolare in cui (2.9) vale con probabilità costante. Il seguente teorema afferma invece che l'algoritmo può essere migliorato in modo che (2.9) valga con probabilità $1 - \delta$.

Teorema 2.8. *Dati una matrice $A \in \mathbb{R}^{m\times n}$, un intero $k \ll \min\{m,n\}$, $\varepsilon \in (0,1]$ e $\delta \in (0,1]$, allora esistono algoritmi randomizzati tali che o vengono scelte esattamente $c = \mathcal{O}(k^2\log(1/\delta)/\varepsilon^2)$ colonne di A per costruire C ed esattamente $r = \mathcal{O}(c^2\log(1/\delta)/\varepsilon^2)$ righe di A per costruire R, o vengono scelte in media $c = \mathcal{O}(k\log k\log(1/\delta)/\varepsilon^2)$ colonne per costruire C ed $r = \mathcal{O}(c\log c\log(1/\delta)/\varepsilon^2)$ per costruire R, in modo che almeno con probabilità $1 - \delta$ valga*

$$\|A - CUR\|_F \leq (1+\varepsilon)\|A - A_k\|_F$$

Dimostrazione. Si esegue l'algoritmo COLUMNSELECT $\ln(2/\delta)$ volte, in modo che ogni esecuzione sia indipendente dall'altra, e tra tutte le matrici ottenute si prende C in modo che $\|A - CC^+A\|_F$ sia il più piccolo. Presa tale C si esegue l'algoritmo ROWSELECT $\ln(2/\delta)$ volte, in modo che ogni esecuzione sia indipendente dall'altra, e tra tutti le matrici ottenute si prendono U ed R in modo che $\|A - CUR\|_F$ sia il più piccolo. Allora si ha che

$$\|A-CUR\|_F \leq (1+\varepsilon)\|A-CC^+A\|_F \leq (1+\varepsilon)^2\|A-A_k\|_F \leq (1+\varepsilon')\|A-A_k\|_F$$

dove $\varepsilon' = 3\varepsilon$, e la probabilità che non valga la disuguaglianza scritta è minore di $(1/e)^{\ln(2/\delta)} + (1/e)^{\ln(2/\delta)} = \delta/2 + \delta/2 = \delta$. Quindi $\|A - CUR\|_F \leq (1 + \varepsilon')\|A-A_k\|_F$ vale almeno con probabilità $1-\delta$. L'algoritmo descritto verifica l'enunciato del teorema e la sua implementazione in Matlab è descritta in Appendice B.6. □

Nonostante garantisca una buona approssimazione della matrice di partenza, nella pratica, l'algoritmo presentato nella dimostrazione precedente è molto costoso poiché, per ogni terna C, U ed R è necessario calcolare le due norme $\|A-CC^+A\|_F$ e $\|A-CUR\|_F$ per stabilire quale sia la scelta migliore tra le matrici ottenute.

Capitolo 3

Applicazioni della decomposizione CUR

La decomposizione CUR viene utilizzata principalmente quando si ha la necessità di individuare delle proprietà o degli elementi che caratterizzano il dataset studiato. Tale decomposizione si comporta molto bene in molte aree dell'analisi dei dati come la classificazione, l'eliminazione del rumore, la ricostruzione di immagini ed il clustering. Tra le numerose ed interessanti possibilità d'impiego, riportiamo nel seguito alcuni risultati numerici ottenuti riducendo le dimensioni di un dataset di documenti (pagine web) mediante le decomposizioni CX e CUR.

Definizione 3.1. Un *dataset* di m oggetti ognuno dei quali è descritto da n caratteristiche è rappresentato da una matrice $A \in \mathbb{R}^{m \times n}$.

Il dataset utilizzato per applicare i metodi presentati in precedenza, è un dataset dell'Open Directory Project (ODP) e contiene 139 documenti (pagine web) provenienti da due diverse categorie della ODP:

- `US:Indiana:Evansville`
- `US:Florida`

Questo dataset, che verrà indicato con A, definisce una matrice *documenti*$\times$ *termini* che ha sulle righe i documenti, mentre sulle colonne i termini chiave dei documenti. Quindi, l'elemento $a_{ij} \in A$ rappresenta il numero di occorrenze del termine j-esimo nel documento i-esimo. I documenti sono stati organizzati nella matrice `A` da Evgeniy Gabrilovich nell'ambito del progetto *Technion Repository of Text Categorization Datasets* (TechTC) ed è possibile scaricarli dal sito `http://techtc.cs.technion.ac.il/techtc100/`.

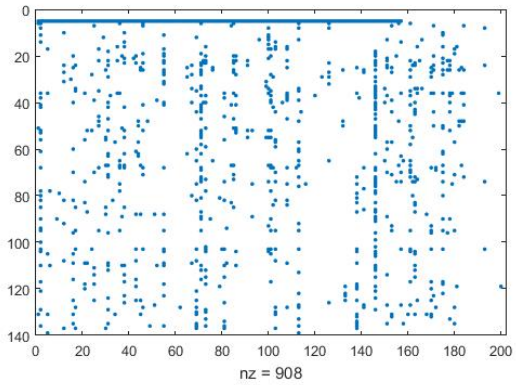

Figura 3.1: Spy della matrice `A(:,800:1000)`

La matrice A è molto grande, $A \in \mathbb{R}^{139\times 15170}$, non negativa e sparsa come si può vedere in Figura 3.1 dove è rappresentato il pattern di sparsità di una porzione di dimensioni 139×200 della matrice A.

3.1 Strategia di riduzione

Per ridurre le dimensioni del dataset A descritto in precedenza, sono stati utilizzati gli algoritmi COLUMNSELECT per la decomposizione CX e ALGORITHMCUR per la decomposizione CUR. A partire dai risultati ottenuti dalle due decomposizioni, è stata calcolata la norma dell'errore (divisa per $\|A - A_k\|_F$ dove A_k è la SVD troncata di rango k) come funzione del numero di colonne c scelte esattamente o in media. Per rappresentare queste quantità nei grafici riportati nel seguito, lavorando con algoritmi randomizzati, invece di rappresentare il risultato di una singola esecuzione, sono stati eseguiti più volte gli algoritmi di decomposizione CX e CUR ed è stata fatta la media di tutti i risultati ottenuti.
Per la scelta delle colonne sono stati utilizzati entrambi i metodi di subspace sampling:

- L'algoritmo EXACTLY(c) che permette di prendere più volte la stessa colonna.

- L'algoritmo EXPECTED(c) che permette di prendere una colonna al più una volta.

La prima quantità studiata è $\Theta_1 = \|A - CC^+A\|_F / \|A - A_k\|_F$ che, per il Teorema 2.5, è maggiorata con una probabilità vicina ad 1 da $(1 + \varepsilon)$. Per $c = k$ la quantità Θ_1 non dovrebbe essere minore di 1, mentre per $c > k$ lo scalare Θ_1 può essere più piccola di 1.
La seconda quantità studiata è $\Theta_2 = \|A - CUR\|_F / \|A - A_k\|_F$ che, per il Teorema 2.7, è maggiorata con una probabilità vicina ad 1 da $(1 + \varepsilon)$. Per calcolare Θ_2 il numero delle righe scelte in media o esattamente, è stato impostato come il doppio delle colonne; lavorando sulla scelta di questo parametro si possono ottenere risultati leggermente migliori di quelle presentati.

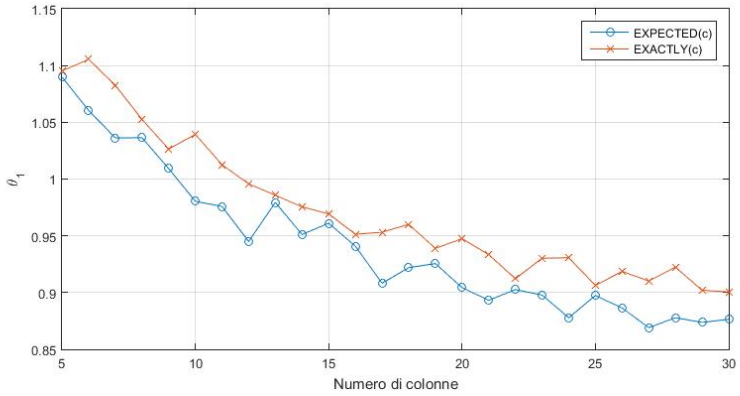

Figura 3.2: Errore decomposizione CX per $k = 5$

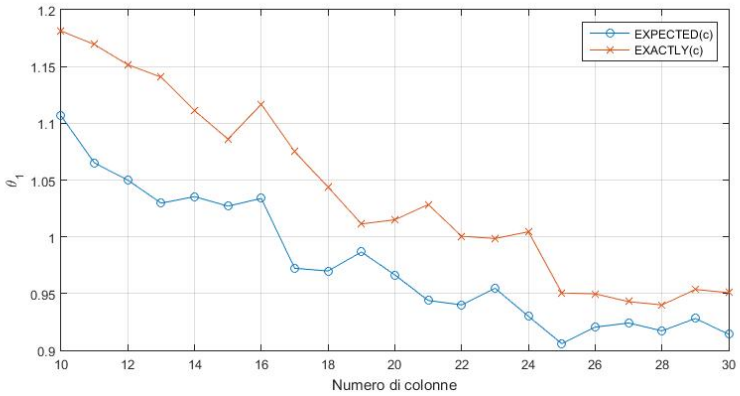

Figura 3.3: Errore decomposizione CX per $k = 10$

In Figura 3.2 è rappresentato Θ_1 al variare di c per $k = 5$. Si può osservare

che Θ_1 diventa più piccolo di 1 solo quando il numero di colonne supera il doppio di k. Allo stesso modo, si può osservare un comportamento simile in Figura 3.3 dove è rappresentato Θ_1 al variare di c per $k = 10$. In entrambi i grafici è evidente che, campionando le colonne con EXPECTED(c), Θ_1 è più piccolo di quello ottenuto utilizzando EXACTLY(c).

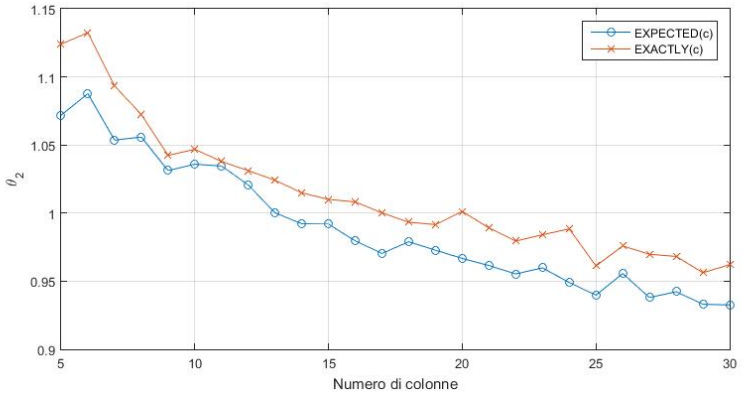

Figura 3.4: Errore decomposizione CUR per $k = 5$

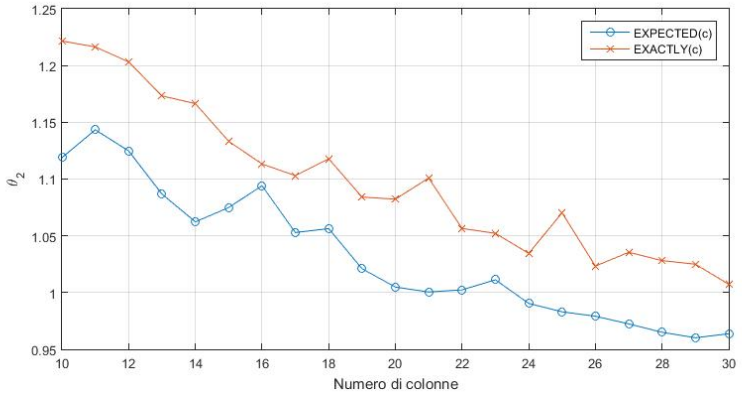

Figura 3.5: Errore decomposizione CUR per $k = 10$

In Figura 3.4 è rappresentato Θ_2 al variare di c per $k = 5$, mentre in Figura 3.5 è rappresentato Θ_2 al variare di c per $k = 10$. Anche in questi grafici si osserva chiaramente che utilizzando EXPECTED(c) invece di EXACTLY(c) si ha un errore Θ_2 più piccolo. Tale differenza è giustificata dal fatto che EXPECTED(c) può campionare un numero di colonne maggiore di c.

Dai risultati ottenuti è evidente che, fissato k, la decomposizione CUR ha un'errore di approssimazione molto vicino a quello della SVD troncata di rango k. Per tale motivo, in molte applicazioni si preferisce utilizzare la decomposizione CUR poiché si ha le necessità di avere un errore "piccolo "ed al contempo interpretare i risultati che si ottengono.

Conclusioni

Sono stati presentati ed analizzati degli algoritmi randomizzati per determinare una decomposizione matriciale di basso rango, che permette di rappresentare la matrice in termini alcune colonne/righe della matrice stessa. Questi algoritmi hanno delle garanzie sull'errore, che derivano dalla particolare scelta delle colonne/righe mediante il subspace sampling. In particolare, la parte centrale dell'analisi dell'errore, segue dalla risoluzione approssimata di un problema di regressione lineare multidimensionale utilizzando strategie di sampling.
Nonostante la decomposizione CUR sia utilizzabile in diversi campi dell'analisi dei dati per benefici che comporta, ci delle questioni aperte alle quali si stanno ancora cercando delle risposte:

- L'unicità della decomposizione.
- La possibilità di estendere la validità del risultato sull'errore ad altre norme matriciali.
- L'esistenza di un algoritmo deterministico per la scelta ottimale delle colonne e/o righe, in modo che l'output dell'algoritmo sia lo stesso per ogni esecuzione.
- L'esistenza di una condizione semplice che permetta, dopo aver scelto le righe e/o le colonne, di determinare se con quel sample è stato raggiunta un'approssimazione con errore vicino a $(1 + \varepsilon)$, senza la necessità di calcolare la norma $\|A - CUR\|_F$.
- La stabilità numerica dell'algoritmo presentato.

Bibliografia

[1] Lars Elden, *Matrix Methods in Data Mining and Pattern Recognition* , SIAM, April 2007.

[2] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Johns Hopkins Press, Baltimore, 1996.

[3] A. Ben-Israel and T.N.E. Greville, *Generalized Inverses: Theory and Applications*, SpringerVerlag, New York, 2003.

[4] G.W. Stewart, *Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix*, Numerische Mathematik, 1999.

[5] G.W. Stewart, *Error analysis of the quasi-Gram-Schmidt algorithm*, Technical Report UMIACS TR-2004-17 CMSC TR-4572, University of Maryland, College Park, MD, 2004.

[6] M.W. Berry, S.A. Pulatova, and G.W. Stewart, *Computing sparse reduced-rank approximations to sparse matrices*, Technical Report UMIACS TR-2004-32 CMSC TR-4589, University of Maryland, College Park, MD, 2004.

[7] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra and Its Applications, 1997.

[8] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method* Computing, 2000.

[9] S.A. Goreinov and E.E. Tyrtyshnikov, *The maximum-volume concept in approximation by low-rank matrices*, Contemporary Mathematics, 2001.

[10] M. Gu and S.C. Eisenstat *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM Journal on Scientific Computing, 1996.

[11] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the approximation of matrices*, Technical Report YALEU/DCS/TR-1361, Yale University Department of Computer Science, New Haven, CT, June 2006.

[12] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, Journal of the ACM, 2004.

[13] P. Drineas, R. Kannan, and M.W. Mahoney, *Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition*, SIAM Journal on Computing, 2006.

[14] Drineas P, Mahoney MW, Muthukrishnan S, *Relative-error CUR matrix decompositions*, SIAM J Matrix Anal Appl, 2008.

Appendice A

Algoritmi sampling

In questa sezione, vengono descritte le due procedure complementari per la campionatura random di colonne e/o righe da una matrice in input A. Entrambi gli algoritmi prendono in input $A \in \mathbb{R}^{m \times n}$ ed una distribuzione di probabilità $\{p_i\}_{i=1}^{n}$, e costruiscono una matrice C formata da copie scalate di alcune colonne di A.
Il primo algoritmo è l'algoritmo EXACTLY(c) che seleziona esattamente c colonne di A, mediante c estrazioni indipendenti identicamente distribuite (i.i.d), dove per ogni estrazione la probabilità che l'i-esima colonna di A venga presa è p_i.

Input $A \in \mathbb{R}^{m \times n}$, $p_i \geq 0$ per $i = 1, \ldots, n$ tali che $\sum_{i=1}^{n} p_i = 1$, un intero positivo $c \leq n$.

Output Una matrice di sampling S ed una matrice diagonale di scaling D e la matrice C formata da alcune colonne di A riscalate.

Inizializzazione S e D come matrici nulle

Per $j = 1, \ldots, c$

- estrazione $i_j \in \{1, \ldots, n\}$ con $P(i_j = i) = p_i$
- $S_{i_j j} = 1$
- $D_{jj} = 1/\sqrt{c p_{i_j}}$

$C = ASD$

Algoritmo EXACTLY(c)

Osservazione. Per costruzione, le colonne della matrice di sampling $S \in \mathbb{R}^{n\times c}$ sono formate da zeri tranne per un elemento, mentre la matrice di scaling $D \in \mathbb{R}^{c\times c}$ è una matrice diagonale.

Il secondo algoritmo è l'algoritmo EXPECTED(c) che in media sceglie al massimo c colonne di A per formare C, inserendo l'i-esima di A in C con una probabilità $\bar{p}_i = \min\{1, cp_i\}$.

Input $A \in \mathbb{R}^{m\times n}$, $p_i \geq 0$ per $i = 1, \ldots, n$ tali che $\sum_{i=1}^{n} p_i = 1$, un intero positivo $c \leq n$.

Output Una matrice di sampling S ed una matrice diagonale di scaling D e la matrice C formata da alcune colonne di A riscalate.

Inizializzazione S e D come matrici nulle

Inizializzazione $t = 1$

Per $j = 1, \ldots, n$

Si prende j con una probabilità pari a $\bar{p}_i = \min\{1, cp_j\}$

Se j viene preso allora

- $S_{jt} = 1$
- $D_{tt} = 1/\min\{1, \sqrt{cp_j}\}$
- $t = t + 1$

Restituisce $C = ASD$

Algoritmo EXPECTED(c)

Osservazione. Per costruzione, le colonne della matrice di sampling $S \in \mathbb{R}^{n\times n}$ sono formate da zeri tranne per un elemento, mentre la matrice di scaling $D \in \mathbb{R}^{n\times c'}$ con $c' \leq c$ (in media) è una matrice che ha elementi non nulli solo nella diagonale principale.

Appendice B

Listati dei programmi

B.1 Listati degli algoritmi di sampling

Con `A` viene indicata una matrice $A \in \mathbb{R}^{m \times n}$, con `pj` un vettore riga contenente le probabilità di campionamento per le colonne, mentre `c` è un intero positivo minore di n. Invece con `S` è la matrice di sampling e `D` è la matrice di scaling.

```
function [S,D]=Exactly(A,pj,c)
    [m,n]=size(A);
    S=zeros(n,c);D=zeros(c,c);
    for t=1:c
      i=datasample([1:length(pj)],1,'Weights',pj);
      S(i,t)=1;
      D(t,t)=1/sqrt(c*pj(i));
    end
end
```

```
function [S,D]=Expected(A,pj,c)
    [m,n]=size(A);
    S=zeros(n,1); D=[]; t=1;
    for i=1:n
      pi=pj(i);
      if rand<min(1,c*pi)
        S(i,t)=1;
        D(t,t)=1/min(1,sqrt(c*pi));
        t=t+1;
      end
    end
end
```

B.2 Funzione ColumnSelect

```
function C=ColumnSelect(A,pj,c,opt)
% A matrice da cui vengono campionate le colonne
% pj vettore delle probabilità di campionamento per le colonne
% c il numero di colonne scelte in media o esattamente
% opt parametro per utilizzare Expected o Exactly

  if opt==1
    [S,D]=Expected(A,pj,c);
  else
    [S,D]=Exactly(A,pj,c);
  end
  C=A*S*D;
```

B.3 Funzione RowSelect

```
function [R,W,U]=RowSelect(A,C,pj,r,opt)
% A matrice da cui vengono campionate le colonne
% C matrice formata da alcune colonne di A
% pj vettore delle probabilità di campionamento per le righe
% r numero di righe scelte in media o esattamente
% opt parametro per utilizzare Expected o Exactly

  if opt==1
    [S,D]=Expected(A',pj,r);
  else
    [S,D]=Exactly(A',pj,r);
  end
  R=D'*S'*A;
  W=D'*S'*C;
  U=pinv(W);
```

B.4 Funzione BetterColumnSelect

```
function [C,res]=BetterColumnSelect(A,pj,c,opt,it)
% A matrice da cui vengono campionate le colonne
% pj vettore delle probabilità di campionamento per le colonne
% c il numero di colonne scelte in media o esattamente
% opt parametro per utilizzare Expected o Exactly
```

```
% it numero di iterazioni

  C=ColumnSelect(A,pj,c,opt);
  res=norm(A-C*pinv(C)*A,'fro');
  for i=1:it
    C1=ColumnSelect(A,pj,c,opt);
    res1=norm(A-C1*pinv(C1)*A,'fro');
    if res1<res
      C=C1;
      res=res1;
    end
  end
```

B.5 Funzione BetterRowSelect

```
function [R,W,U,res]=BetterRowSelect(A,C,k,r,opt,it)
% A matrice da cui vengono campionate le colonne
% C matrice formata da alcune colonne di A
% pj vettore delle probabilità di campionamento per le righe
% r numero di righe scelte in media o esattamente
% opt parametro per utilizzare Expected o Exactly
% it numero di iterazioni

  [R,W,U]=RowSelect(A,C,pi,r,opt);
  res=norm(A-C*U*R,'fro');
  for i=1:it
    [R1,W1,U1]=RowSelect(A,C,pi,r,opt);
    res1=norm(A-C*U1*R1,'fro');
    if res1<res
      res=res1;
      R=R1;
      W=W1;
      U=U1;
    end
  end
```

B.6 Funzione `AlgorithmCUR`

Per ottimizzare i costi computazionali, per implementare ALGORITHMCUR si è preferito calcolare la SVD troncata e le probabilità di campionamento solamente all'interno di ALGORITHMCUR.

```
function [C,U,R,res]=AlgorithmCUR(A,k,c,r,opt,it)
% A matrice da cui vengono campionate le colonne
% k parametro di rango
% c numero di colonne scelte in media o esattamente
% r numero di righe scelte in media o esattamente
% opt parametro per utilizzare Expected o Exactly
% it numero di iterazioni

   [U,S,V]=svds(A,k);
   pj=sum(V.*V,2)/k;
   pi=sum(U.*U,2)/k;
   [C,resc]=BetterColumnSelect(A,pj,c,opt,it);
   [R,W,U,res]=BetterRowSelect(A,C,pi,r,opt,it);
```