Alma Mater Studiorum · Università di Bologna

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

APPROSSIMAZIONE DI MATRICI ED APPLICAZIONE AL TEXT MINING

Tesi di Laurea in Analisi Numerica

Relatore:
Chiar.ma Prof.ssa
VALERIA SIMONCINI

Presentata da:
DAVIDE PALITTA

II Sessione
Anno Accademico 2011/2012

Al mio papà.

Indice

Capitolo 1

Preambolo

1.1 Introduzione

Questa tesi ha come obbiettivo quello di presentare alcuni metodi di approssimazione mediante fattorizzazione di matrici. Alcuni di questi sono classici mentre altri più recenti. In particolare presenteremo:

- Svd troncata;
- Metodo di Lanczos;
- Fattorizzazione non negativa.

Applicheremo poi questi metodi a matrici non negative. Questa scelta è dettata dal fatto che molto spesso, nelle applicazioni, i dati con cui si lavora hanno valori appunto non negativi. Alcuni importanti esempi sono:

- In una raccolta di immagini, queste possono essere rappresentate da vettori le cui componenti corrispondono ai singoli pixel. L'intensità e il colore del pixel vengono indicati con un numero non negativo andando a formare così una matrice non negativa $pixel \times immagini$.
- Nella raccolta di documenti questi sono archiviati come vettori. Le loro componenti *contano* le occorrenze con cui un termine appare in quel determinato documento. Concatenando i vettori dei documenti uno dopo l'altro si crea così una matrice $termini \times documenti$ che rappresenta numericamente l'intera collezione.

In questa tesi applicheremo i metodi sopra citati nell'ambito del *text mining*. Con questo termine si indicano tutti quei metodi usati per estrarre informazioni utili da una grande e spesso non strutturata collezione di testi. Una applicazione tipica è la ricerca di articoli scientifici in un grande database. Ad esempio, in ambito medico, si vogliono trovare tutti gli articoli che parlano di una determinata malattia all'interno di un database. Allora si inizia costituendo una frase di ricerca (*query*) contenente le parole-chiave che si ritengono più opportune. Il sistema quindi confronta la query con i documenti del database e presenta all'operatore i documenti più rilevanti.

Tra i tre principali obbiettivi della fattorizzazione di queste matrici abbiamo:

1. Raggruppare oggetti simili della collezione in gruppi (*clustering*);
2. Ritrovare, all'interno della collezione, gli oggetti desiderati;
3. Identificare strutture interpretabili all'interno della collezione.

Fino a qualche anno fa la tecnica più usata in quest'ambito era la ben nota Latent Semantic Indexing (LSI) che sfrutta una fattorizzazione della matrice dei dati creandone una approssimazione di rango basso. Questa fattorizzazione è la Singular Value Decomposition (SVD) molto efficace per ottenere i punti 1. e 2. sopra ma non per il 3. Infatti la SVD non fornisce all'utente alcuna interpretazione dei fattori in gioco, non rivela nulla sulla collezione dei dati. Tale problema ha stimolato lo studio di fattorizzazioni alternative, che permettono di trattare il punto 3. in modo soddisfacente. Una di queste è la fattorizzazione non negativa (Nonnegative Matrix Factorization - NMF). Si mostrerà in questa tesi che la NMF è sufficientemente accurata per ottenere i punti 1. e 2., senza raggiungere l'accuratezza della SVD, ma che permette una migliore interpretazione dei dati in accordo con il punto 3.

I metodi di fattorizzazione applicati alla matrice dei dati A non negativa, ci permetteranno di averne una approssimazione di rango basso. Questo porta numerosi vantaggi, infatti l'approssimazione di rango k di A, indicata con A_k, spesso permette un risparmio di memoria ma soprattutto fornisce una rappresentazione più pulita e più efficiente delle relazioni che intercorrono tra i dati. L'approssimazione di rango basso identifica le componenti essenziali dei dati ignorando quelle superflue o attribuibili a rumore e inconsistenza.

1.2 Notazioni

Come abbiamo detto, in molte applicazioni, ritroviamo matrici non negative. Diamone quindi una definizione più formale:

Definizione 1.1 (Matrice Non Negativa)**.** Sia A una martrice $m \times n$ a coefficienti reali. Allora A è una **matrice non negativa** se tutti i suoi elementi sono non negativi, cioè se

$$a_{ij} \geq 0 \qquad \forall i = 1, \ldots, m, \forall j = 1, \ldots, n.$$

Altre definizioni che useremo all'interno di questa tesi sono le seguenti.

Definizione 1.2 (Matrice Ortogonale)**.** Sia $A \in \mathbb{R}^{n \times n}$. Allora dico che A è **ortogonale** se

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

In particolare quindi se A è ortogonale significa che 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 è *unitaria* se

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

dove con A^* indico la trasposta coniugata di A. Ciò significa che se A è ortogonale allora è anche unitaria.

Definizione 1.3 (Matrice Sparsa e Piena)**.** Sia $A \in \mathbb{R}^{n \times m}$. Allora si dice che A è **sparsa** se per ogni riga di A, gli elementi non nulli sono solo il $3 - 5\%$. In altre parole $\forall j = 1, \ldots, m$, $a_{ij} = 0$ per quasi tutti gli indici $i = 1, \ldots, n$.
Se A non è sparsa allora è detta **piena**.

Definizione 1.4 (Norma-1 e Norma-2 di vettori)**.** Sia $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$. Allora si definisce la **norma-1** di x come segue:

$$||x||_1 = \sum_{i=1}^{n} |x_i|.$$

Si definisce invece la **norma-2** o norma euclidea di x come

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

Definizione 1.5 (Norma-2 e Norma di Frobenius di matrici)**.** Sia $A \in \mathbb{R}^{n\times m}$. Allora si definisce la **norma-2** di A come segue:

$$||A||_2 = \max_{\substack{x\in\mathbb{R}^m \\ x\neq 0}} \frac{||Ax||_2}{||x||_2}.$$

Questa norma è anche detta norma euclidea indotta poichè si utilizza la norma euclidea vettoriale. Si definisce invece, la **norma di Frobenius** come:

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

Capitolo 2

SVD: Decomposizione ai Valori Singolari

2.1 La SVD completa

La Singular Value Decomposition (SVD) o Decomposizione ai Valori Singolari, consiste nel rappresentare una matrice data A nel prodotto di altre tre matrici di cui due ortogonali ed una diagonale.

Teorema 2.1.1 (Teorema di 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 ortogonali e $\Sigma \in \mathbb{R}^{n\times n}$ é diagonale:

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

con $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ detti valori singolari.

In particolare le colonne di $U = [u_1, \ldots, u_n]$ sono dette *vettori singolari sinistri* mentre quelle di $V = [v_1, \ldots, v_n]$ *vettori singolari destri.* La decomposizione ai valori singolari di A permette di scrivere quest'ultima come $A = \sum_{i=1}^{n} u_i \sigma_i v_i^T$ dove $u_i \sigma_i v_i^T$ è detta i-esima componente singolare di A.

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

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

e sia $\bar{x}$ il vettore soluzione. Sia

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

con $||y||_2 = 1$ dove $\sigma_1 = ||A||_2$. Si definiscano 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 entrambe ortogonali. Quindi

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

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

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

poichè A_1 è ottenuta da A mediante trasformazioni ortogonali. Sia

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

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

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

Quindi

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

Siccome $\sigma_1 = \max \frac{||Ax||}{||x||}$ deve essere $d = 0$.

Quindi sia la prima riga che la prima colonna di A_1 sono zero, eccetto che per l'elemento diagonale. La procedura prosegue in modo iterativo con B. Alla fine si avrà $U = Y_1 \cdots Y_{n-1}$ e $V = X_1 \cdots X_{m-1}$. □

2.2 La SVD troncata

Come abbiamo già visto, la decomposizione ai valori singolari della matrice data A 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 racchiuse le k componenti singolari principali di A. Nell'ambito del *text mining*, i valori singolari della matrice dei dati A hanno spesso un caratteristico andamento a gomito:

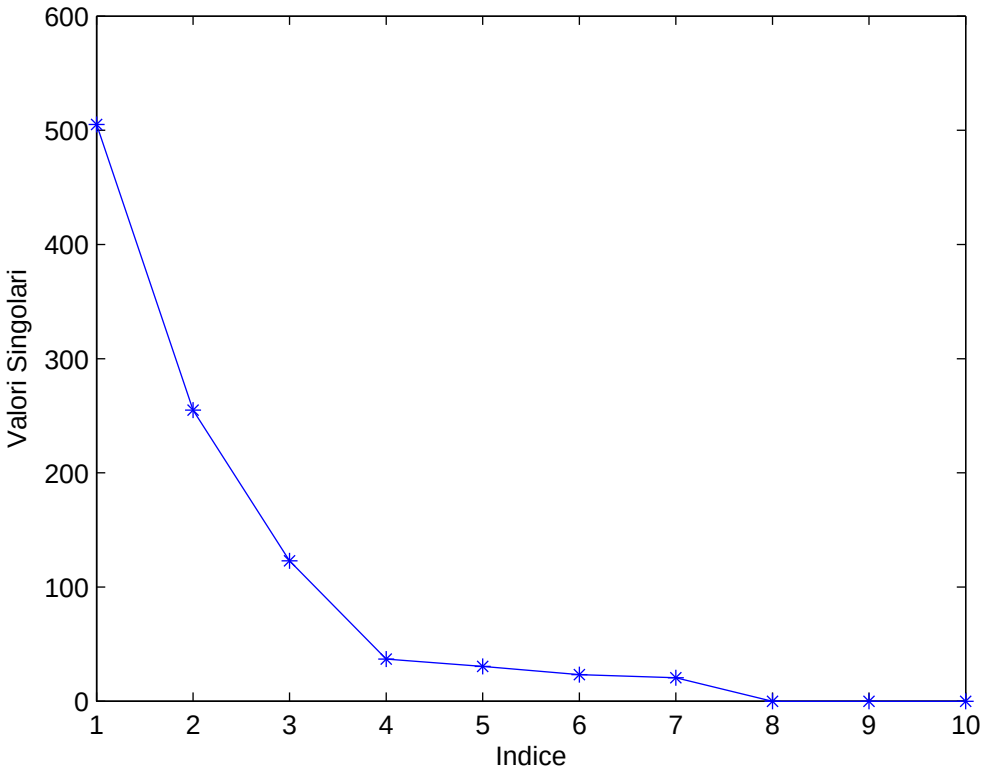

Figura 2.1: Andamento a gomito dei Valori Singolari

In questo modo si possono stimare facilmente le prime k componenti principali di A.

La svd troncata consiste nell'approssimare A proprio con le sue prime k componenti principali

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

interpretando quindi la matrice N come rumore.

Nell'ambito del *text mining*, la tecnica che lavora con A_k invece che con A è chiamata Latent Semantic Indexing (LSI) perchè l'approssimazione di rango basso rivela connessioni tra i dati che erano nascoste o, appunto, latenti nella matrice A.

2.3 Caratterizzazione della SVD troncata

Matematicamente, la SVD troncata ha una importante proprietà:

Teorema 2.3.1. *Sia $A \in \mathbb{R}^{m\times n}$ una matrice di rango $r > k$. Allora il problema di minimo*

$$\min_{rango(Z)=k} ||A - Z||_2$$

con $||.||_2$ norma-2, ha come soluzione

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

dove A_k è la decomposizione ai valori singolari di rango k di A.
Inoltre vale

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

Vale anche il corrispondente teorema dove viene usata la norma di Frobenius.

Teorema 2.3.2. *Sia $A \in \mathbb{R}^{m\times n}$ una matrice di rango $r > k$. Allora il problema di minimo*

$$\min_{rango(Z)=k} ||A - Z||_F$$

con $||.||_F$ norma di Frobenius, ha come soluzione

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

dove A_k è la decomposizione ai valori singolari di rango k di A. Inoltre vale

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

Dimostrazione. Sia dato il seguente prodotto scalare definito mediante la norma di Frobenius

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

con $A, B \in \mathbb{R}^{m\times n}$. Allora si nota che le matrici

$$u_i v_j^T \quad \forall i = 1,\dots,m \quad \forall j = 1,\dots,n$$

costituiscono una base ortonormale di $\mathbb{R}^{m\times n}$ rispetto a questo prodotto scalare. Infatti

$$\left\langle u_i v_j^T, u_k v_l^T \right\rangle = \mathrm{tr}(v_j u_i^T u_k v_l^T) = \mathrm{tr}(v_l^T v_j u_i^T u_k) = (v_l^T v_j)(u_i^T u_k).$$

Quindi le matrici $u_i v_j^T$ sono ortonormali e, essendo mn in numero, costituiscono una base di $\mathbb{R}^{m\times n}$. Scriviamo allora la matrice $Z \in \mathbb{R}^{m\times n}$ in termini di questa base

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

dove si devono calcolare i coefficienti ζ_{ij}. A causa dell'ortonormalità della base considerata si ha

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

dove con σ_{ij} si indicano gli elementi della matrice Σ cioè i valori singolari della matrice A. Ovviamente si può scegliere la matrice Z in modo che questa sia diagonale e che quindi il secondo addendo dell'equazione sopra sia uguale a zero. Si ha quindi:

$$Z = \sum_{i} \zeta_{ii} u_i v_i^T.$$

Dato che il rango della matrice Z è uguale al numero di elementi non nulli di questa somma, imporre $\mathrm{rango}(Z) = k$ implica che all'interno della somma debbano esserci esattamente k termini non nulli. Per raggiungere il minimo di questa funzione si deve scegliere $\zeta_{ii} = \sigma_{ii}, \quad i = 1,\dots,k$, che porta alla tesi, infatti

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

da cui

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

□

I precedenti teoremi mostrano che la SVD troncata fornisce un termine di paragone rispetto al quale possiamo valutare l'accuratezza di tutte le altre approssimazioni di rango basso. Questa proprietà è molto utile anche a livello pratico: gli algoritmi per ricavare le k componenti singolari più significative sono accurati, ben definiti e robusti.

Un altro vantaggio della SVD troncata consiste nella costruzione di una successione di approssimazioni di rango basso nel senso che, una volta calcolata A_k, si ha a disposizione

$$A_j \quad \forall j \leq k$$

senza nessun calcolo aggiuntivo. In ogni caso la SVD troncata ha qualche svantaggio. Generalmente, soprattutto nelle applicazioni, la matrice dei dati A è molto sparsa e applicandole la SVD troncata vengono sempre prodotte delle componenti singolari che sono quasi completamente dense. Questo significa che, in molti casi, memorizzare A_k, con k grande e A sparsa, cioè memorizzare i vettori $u_i, v_i, \quad i = 1, \ldots, k$, richiede più (spesso molto più) spazio che memorizzare A.

In più, come abbiamo già detto, nell'ambito del *text mining*, A è sempre una matrice non negativa mentre le componenti singolari create con la SVD troncata hanno anche segno negativo. Perdere la struttura di non negatività della matrice dei dati fa sì che i fattori della SVD troncata non siano interpretabili.

Capitolo 3

Il metodo di Lanczos

3.1 Approssimazione di autocoppie

Il metodo di Lanczos [6] nasce come metodo per l'approssimazione di autocoppie di matrici. Data $A \in \mathbb{R}^{n \times n}$ simmetrica, siano $(\lambda_i, v_i) \quad i = 1, \ldots, n$ le autocoppie di A, cioè

$$Av_i = \lambda_i v_i \quad \text{con} \quad ||v_i|| = 1, \qquad \forall i = 1, \ldots, n.$$

Si vogliono determinare

$$(\bar{\lambda}_i, \bar{v}_i) \qquad i = 1 \ldots n$$

tali che

$$\frac{||A\bar{v}_i - \bar{\lambda}_i \bar{v}_i||}{||\bar{\lambda}_i||} < tol$$

dove con *tol* si indica una tolleranza prefissata. Nel metodo di Lanczos viene costituito uno spazio K di dimensione m ed una sua base ortonormale $\{q_1, \ldots, q_m\}$ e si vogliono determinare i vettori

$$\bar{v}_i = Q_m x_i \quad \text{con} \quad ||\bar{v}|| = 1, \qquad i = 1, \ldots, m,$$

con $Q_m = [q_1, \ldots, q_m]$, e tali che

$$\bar{v}_i \approx v_i \quad \forall i = 1, \ldots, m.$$

Una volta trovato l'i-esimo autovettore approssimante normalizzato, si avrà automaticamente anche l'autovalore approssimante associato

$$\bar{\lambda}_i = {\bar{v}_i}^T A \bar{v}_i \approx \lambda_j \qquad j \in \{1, \ldots, n\}.$$

In Matlab, il metodo di Lanczos viene implementato all'interno della funzione `eigs`. Questa funzione è utilizzata per calcolare alcuni degli autovalori e degli autovettori associati, di una matrice sparsa A. La funzione verifica che A sia simmetrica, se lo è, allora applica il metodo di Lanczos.

Quindi, ricapitolando, i primi passi del metodo di Lanczos consistono in:

- Scelta dello spazio approssimante K;
- Scelta dei vettori $\bar{v_i} = Q_m x_i \in K$.

3.2 Il sottospazio di Krylov

Una scelta particolarmente appropriata per lo spazio approssimante è il sottospazio di Krylov così definito:

Definizione 3.1 (Sottospazio di Krylov)**.** Sia $A \in \mathbb{R}^{n\times n}$ e sia $m \in \mathbb{N}$. Si scelga a priori $v \in \mathbb{R}^n$ con $v \neq 0$, allora si definisce il sottospazio di Krylov di dimensione m generato da A e v il seguente sottospazio:

$$K_m(A, v) := \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\}.$$

Questo sottospazio gode di numerose e importanti proprietà, alcune delle quali sono le seguenti:

1. $K_m(A, v) \subset K_{m+1}(A, v)$ e $AK_m(A, v) \subset K_{m+1}(A, v)$;
2. Esiste un $\overline{m} \leq n$ con n ordine di A tale che

   $$K_{\overline{m}}(A, v) = K_{\overline{m}+1}(A, v)$$

   cioè lo spazio $K_{\overline{m}}(A, v)$ è invariante per A;
3. Rango$(K_m(A, v)) \leq m$;
4. Ogni vettore $w \in K_m(A, v)$ si può scrivere come

   $$w = p(A)v$$

   con p polinomio di grado minore di m;
5. Se $\{q_1, \ldots, q_m\}$ è una base ortonormale di $K_m(A, v)$, allora

   $$K_{m+1}(A, v) = \text{span}\{q_1, \ldots, q_m, A^m q_1\} = K_m(A, v) \cup \{q_{m+1}\}$$

La procedura seguita da Lanczos per determinare una base ortonormale $\{q_1, \ldots, q_m\}$ di $K_m(A, v)$ è la seguente. Fissato $v \in \mathbb{R}^n$:

1. $q_1 = \dfrac{v}{\| v \|}$;
2. Per ogni m:
   (a) $\overline{q}_{m+1} = Aq_m$;
   (b) q_{m+1} è il prodotto dell'ortogonalizzazione di $\overline{q}_{m+1}$ rispetto a $\{q_1, ..., q_m\}$ (mediante il metodo di Gram-Schmidt).

Sia $Q_m = [q_1, \ldots, q_m]$, introduciamo la matrice T_m definita come

$$T_m := Q_m^T A Q_m.$$

Questa matrice è simmetrica, infatti conserva la struttura simmetrica di A, e inoltre è tridiagonale poichè, per costruzione, vale

$$q_i \perp K_m(A, v) \qquad \forall i > m$$

e, sempre per costruzione, $Aq_j \in K_{j+1}(A, v)$. Quindi

$$q_i^T A q_j = q_i^T (A q_j) = 0 \qquad \forall i > j + 1.$$

Per simmetria questo risultato vale anche per ogni $j < i - 1$. Allora

$$T_m = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \end{pmatrix}.$$

Teorema 3.2.1 (Ricorrenza a tre termini)**.** *Per ogni $m > 0$ vale*

$$AQ_m = Q_m T_m + [0, ..., 0, q_{m+1}\beta_m]$$

con

$$T_m = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \end{pmatrix}$$

o, in modo equivalente, vale una ricorrenza a tre termini

$$Aq_i = q_{i-1}\beta_{i-1} + q_i\alpha_i + q_{i+1}\beta_i \qquad \forall i \leq m.$$

Dimostrazione. Per definizione abbiamo

$$\alpha_i = q_i^T A q_i \qquad \text{e} \qquad \beta_i = q_{i+1}^T A q_i.$$

Allora per $i = 1$, l'ortogonalizzazione di Gram-Schmidt implica che:

$$\bar{q}_2 = Aq_1 - q_1(q_1^T A q_1) \equiv Aq_1 - q_1\alpha_1,$$

da cui

$$q_2 = \frac{\bar{q}_2}{||\bar{q}_2||}.$$

Si nota anche che

$$\beta_1 = q_2^T A q_1 = \frac{\bar{q}_2}{||\bar{q}_2||}(\bar{q}_2 + q_1\alpha_1).$$

Per costruzione $\bar{q}_2 \perp q_1$, allora

$$\beta_1 = \frac{\bar{q}_2}{||\bar{q}_2||}\bar{q}_2 = ||\bar{q}_2||.$$

Quindi

$$Aq_1 = [q_1, q_2] \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}.$$

Infatti

$$\bar{q}_2 = Aq_1 - q_1\alpha_1.$$

D'altra parte $\bar{q}_2 = \beta_1 q_2$, da cui $\beta_1 q_2 = Aq_1 - q_1\alpha_1$, e quindi

$$Aq_1 = q_1\alpha_1 + \beta_1 q_2 = [q_1, q_2] \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}.$$

Per i vettori successivi si prosegue in modo analogo. □

3.3 La condizione di Galerkin

Supponiamo di aver determinato, con il metodo di Lanczos, lo spazio $K_m(A, v)$. Si vuole quindi determinare il vettore

$$\bar{v} = Q_m x \in K_m(A, v), \qquad ||\bar{v}||_2 = 1.$$

Allora per calcolare il vettore colonna dei coefficienti $x = (x_1, \ldots, x_m)^T$ imponiamo che il residuo soddisfi la condizione di Galerkin:

$$r := A\bar{v} - \bar{v}(\bar{v}^T A\bar{v}) \perp K_m(A, v)$$

cioè che il vettore residuo sia ortogonale allo spazio stesso. Si nota che questa condizione equivale alla seguente

$$Q_m^T r = 0 \Leftrightarrow Q_m^T A Q_m x - Q_m^T Q_m x\bar{\lambda} = 0$$

dove è stato posto $\bar{\lambda} = \bar{v}^T A\bar{v} = x^T Q_m^T A Q_m x$. Quindi si ha

$$T_m x = \bar{\lambda} x \quad e \quad \bar{\lambda} = x^T T_m x$$

cioè $(\bar{\lambda}_i, x_i) \quad i = 1, \ldots m$ sono le autocoppie di T_m.
Ció significa che i vettori x_i con i quali si approssimano i vettori

$$\bar{v}_i = Q_m x_i \approx v_i$$

non sono altro che gli autovettori della matrice T_m. Diamo quindi la seguente definizione.

Definizione 3.2 (Vettori e Valori di Ritz)**.** Siano $(\bar{\lambda}_i, x_i), \quad i = 1, \ldots m$ le autocoppie della matrice T_m. Allora i vettori

$$\bar{v}_i = Q_m x_i$$

si dicono **vettori di Ritz** mentre gli scalari $\bar{\lambda}_i$ sono detti **valori di Ritz**. Si ha quindi che

$$\bar{v}_i \approx v_j \quad e \quad \bar{\lambda}_i \approx \lambda_j \qquad j \in \{1, \ldots, n\}$$

Si nota anche che il residuo

$$r_i = A\bar{v}_i - \bar{\lambda}_i \bar{v}_i,$$

con $i = 1, \ldots, m$ ha norma

$$\begin{aligned} ||r_i|| &= ||A\bar{v}_i - \bar{\lambda}_i\bar{v}_i|| = ||AQ_m x_i - Q_m x_i\bar{\lambda}_i|| = \\ &= ||(AQ_m x_i - Q_m T_m)x_i|| = ||q_{m+1}\beta_m e_m^T x_i|| = \beta_m |e_m^T x_i|. \end{aligned}$$

Questo significa che è possibile monitorare la convergenza del metodo senza calcolare esplicitamente i vettori di $\mathbb{R}^n$, $\bar{v}_i$ e r_i.

3.4 Il metodo di Lanczos per l'approssimazione di matrici

Come abbiamo già detto, il metodo di Lanczos nasce come metodo per l'approssimazione di autocoppie di matrici. Il metodo di Lanczos però, può essere anche usato come metodo per l'approssimazione della decomposizione ai valori singolari e quindi per l'approssimazione di matrici. Infatti, applicando il metodo alla matrice

$$\mathcal{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

e quindi approssimando le autocoppie di $\mathcal{A}$, si ottengo i valori singolari di A. Infatti, in generale, data una matrice $B \in \mathbb{R}^{n\times m}$ vale che

$$\sigma_i(B) = \sqrt{\lambda_i(BB^T)} \qquad \forall i$$

dove con $\sigma_i(C)$ si indica l'i-esimo valore singolare della matrice C mentre con $\lambda_i(C)$ il suo i-esimo autovalore.
Nel nostro caso dunque sia

$$\lambda_i(\mathcal{A}) = \bar{\lambda}.$$

Allora esiste $v = (x, y)^T \in \mathbb{R}^{2m}$ tale che

$$\mathcal{A}v = \bar{\lambda}v \qquad \Rightarrow \qquad \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \bar{\lambda} \begin{bmatrix} x \\ y \end{bmatrix},$$

da cui

$$\begin{cases} Ay = \bar{\lambda}x \\ A^Tx = \bar{\lambda}y \end{cases}$$

Moltiplicando a sinistra la prima equazione del sistema per A^T si ottiene

$$A^TAy = \bar{\lambda}A^Tx$$

ma $A^Tx = \bar{\lambda}y$, allora

$$A^TAy = \bar{\lambda}^2y.$$

Questo significa che

$$\bar{\lambda}^2 = \lambda_i(A^TA) \qquad \Rightarrow \qquad \bar{\lambda}^2 = \sigma_i(A)^2$$

Quindi

$$\sigma_i(A) = +\sqrt{\lambda_i(A^TA)} \quad \text{e} \quad \sigma_{n-i+1}(A) = -\sqrt{\lambda_i(A^TA)}.$$

Si può inoltre notare che scegliendo un vettore iniziale della forma

$$v = q_1 = \begin{bmatrix} 0 \\ v_1 \end{bmatrix},$$

il metodo di Lanczos produrrà i vettori della base $\{q_1, \ldots, q_m\}$ nel modo seguente:

$$\bar{q}_2 = \mathcal{A}q_1 = \begin{bmatrix} Av_1 \\ 0 \end{bmatrix} \quad \Rightarrow \quad q_2 = ... = \begin{bmatrix} u_1 \\ 0 \end{bmatrix},$$

$$\bar{q}_3 = \mathcal{A}q_2 = \begin{bmatrix} 0 \\ A^T u_1 \end{bmatrix} \quad \Rightarrow \quad q_3 = ... = \begin{bmatrix} 0 \\ v_2 \end{bmatrix},$$

e così fino a q_m.

In questo modo la matrice Q_k con $1 \leq k \leq m$ conterrà i vettori u_j,v_j e la matrice tridiagonale T_k sarà della forma

$$T_k = \begin{pmatrix} 0 & \alpha_1 & & & \\ \alpha_1 & 0 & \beta_2 & & \\ & \beta_2 & 0 & \alpha_2 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}.$$

Questa scelta del vettore iniziale riduce la complessità del metodo di Lanczos a quella della bidiagonalizzazione di Golub-Kahan. Questo metodo consiste nel costruire due matrici V_k e U_{k+1} con colonne ortogonali

$$V_k V_k^T = I_k \qquad U_{k+1} U_{k+1}^T = I_{k+1}$$

e tali che

$$AV_k = U_{k+1} B_k \qquad A^T U_k = V_k B_k^T$$

dove B_k è una matrice bidiagonale superiore

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbb{R}^{k+1 \times k}.$$

3.5 Considerazioni computazionali

Un grande vantaggio del metodo di Lanczos è quello di dare la possibilità di scegliere, a priori, il numero di valori singolari che si vogliono calcolare. Infatti il numero k di valori singolari di A che si vogliono calcolare corrisponde alla dimensione del sottospazio di Krylov $K_k(A, v)$ costruito dal metodo. La SVD troncata invece, calcola necessariamente tutti i valori singolari della matrice A, considerando, solo a posteriori, i primi k più significativi, portando quindi ad un costo computazionale più elevato.

Capitolo 4

La fattorizzazione non negativa

4.1 Caratteristiche della NMF

L'ultimo metodo di approssimazione di matrici presentato in questa tesi è la fattorizzazione non negativa (NMF). L'obiettivo di questo metodo è fattorizzare una matrice data $A \in \mathbb{R}^{m\times n}$ in due fattori non negativi $W \in \mathbb{R}^{m\times k}$, $H \in \mathbb{R}^{k\times n}$ soluzioni del problema di minimo

$$\min_{\substack{W\in\mathbb{R}^{m\times k}, H\in\mathbb{R}^{k\times n}\\ W\geq 0, H\geq 0}} \| A - WH \|_F, \tag{4.1}$$

dove con k si indica il rango dell'approssimazione, che è definito a priori. Si nota che questo problema di minimo è differente dal tipico problema ai minimi quadrati

$$\min_{x\in\mathbb{R}^n} ||b - Ax||_2 \tag{4.2}$$

poichè nel problema (4.1) le variabili W e H non sono vettori bensì matrici. Si mostra però come la norma di Frobenius permetta di ricondurre il problema (4.1) a n problemi della forma (4.2). Siano infatti $B \in \mathbb{R}^{m\times n}, A \in \mathbb{R}^{m\times k}$ e sia

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

un generico problema di minimo della forma (4.1). Allora

$$||B - AX||_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} |b_{ij} - (AX)_{ij}|^2 = \sum_{i=1}^{n}\left(\sum_{j=1}^{m} |b_{ij} - (AX)_{ij}|^2\right) = \sum_{i=1}^{n} ||b_i - Ax_i||_2^2$$

dove con b_i, x_i si indicano rispettivamente l'i-esima colonna di B e l'i-esima colonna di X mentre con $(AX)_{ij}$ la componente di posto ij della matrice AX. Ciò significa che la risoluzione del problema (4.1) equivale alla risoluzione di n problemi della forma (4.2),

colonna per colonna. Matematicamente, il problema (4.1) è non lineare e non convesso in W e H e quindi di difficile trattazione numerica. D'altra parte, è possibile determinarne una approssimazione lineare facendolo diventare così un problema computazionalmente più accessibile. Infatti, considerando alternativamente W oppure H come matrice incognita, si ottiene una dipendenza lineare nella sola W o H. Nasce così il metodo dei minimi quadrati alternati (Alternating Least Squares - ALS). L'algoritmo di base avrà quindi la seguente struttura.

Algoritmo:

1. Inizializzazione di $W^{(1)} \in \mathbb{R}^{m\times k}$;
2. Per $j = 1, 2, ...$ fino a convergenza:
   (a) Risolvere
   $$\min_{H\in\mathbb{R}^{k\times n}, H\geq 0} \| A - W^{(j)}H \|_F \tag{4.3}$$
   ottenendo $H^{(j)}$;
   (b) Risolvere
   $$\min_{W\in\mathbb{R}^{m\times k}, W\geq 0} \| A - WH^{(j)} \|_F$$
   ottenendo $W^{(j+1)}$.

Sia $H = [h_1, \ldots, h_n]$, allora alla k-esima iterazione, la risoluzione dei problemi di minimo matriciali della forma (4.3) in norma di Frobenius corrisponde, come si è già mostrato, alla risoluzione dei problemi vettoriali in norma-2:

$$\min_{\substack{h_j\in\mathbb{R}^k \\ h_j\geq 0}} ||a_j - W^{(k)}h_j||_2, \qquad j = 1, \ldots, n.$$

La fattorizzazione non negativa è preferita ad altri metodi di approssimazione grazie a due caratteristiche fondamentali: interpretabilità e basso utilizzo di memoria che non aumentano i costi computazionali del metodo. Ad esempio, nell'ambito del *text mining*, si vuole fattorizzare la matrice $A \in \mathbb{R}^{m\times n}$ dei *termini*$\times$*documenti*.

In questo caso k può essere considerato come il numero degli argomenti (nascosti) più rilevanti trattati in effetti dalla collezione di documenti. Allora la matrice $W \in \mathbb{R}^{m\times k}$ diventa una matrice *termini*$\times$*argomenti* le cui colonne sono i vettori della base generata dalla NMF. Quindi gli elementi della j-esima colonna di W, indicata con w_j, corrispondono a particolari termini della collezione. Considerando le componenti maggiori di w_j è possibile assegnare un'etichetta al j-esimo vettore della base. Una interpretazione simile può essere data alla matrice $H \in \mathbb{R}^{k\times n}$ considerata come matrice *argomenti*$\times$*documenti* con colonne sparse e non negative. Il j-esimo elemento della i-esima colonna di H indica in che misura il j-esimo argomento è presente nell'i-esimo documento.

D'altra parte però, si ha anche qualche svantaggio. A differenza della SVD, la NMF non è supportata da una base teorica che ne assicura la convergenza ad un minimo assoluto ma gli algoritmi della fattorizzazione non negativa portano, quando c'è convergenza, solo a minimi locali che quindi sono legati, anche in maniera importante, al punto di inizializzazione.

4.2 Algoritmi ACLS e AHCLS

Gli algoritmi ACLS e AHCLS sono delle varianti più sofisticate del metodo ALS che generano, grazie a determinati aggiustamenti, delle matrici sparse che, come abbiamo già detto, sono l'ideale per interpretabilità e bassa necessità di memoria.

Siano a_j e h_j le colonne rispettivamente di A e di H, allora l'algoritmo ACLS (Alternating Constrained Least Squares) consiste nel risolvere, ad ogni iterazione, il seguente problema di minino:

$$\forall j \qquad \min_{h_j} ||a_j - Wh_j||_2^2 + \lambda_H ||h_j||_2^2, \qquad \lambda_H \geq 0, h_j \geq 0, \tag{4.4}$$

La differenza tra il metodo ALS standard e l'algoritmo ACLS consiste nell'aggiunta del fattore

$$\lambda_H ||h_j||_2^2,$$

che va interpretato come un fattore di "penalizzazione" nei riguardi della densità della matrice H. Questo problema infatti, viene anche chiamato problema ai minimi quadrati alternato penalizzato.

Ovviamente la struttura alternante del metodo ALS rimane anche in questa nuova implementazione: nell'iterazione successiva si risolverà il problema di minimo

$$\forall j, \qquad \min_{w'_j} ||a'_j - w'_j H||_2^2 + \lambda_W ||w'_j||_2^2, \tag{4.5}$$

dove con a'_j e w'_j si indicano rispettivamente la j-esima riga di A e la j-esima riga di W.

Esistono algoritmi implementati ad-hoc per la trattazione del vincolo di non negativà dei vettori h_j e w_j, un esempio comune è il Nonnegative Least Square (NNLS) di Lawson e Hanson [5] implementato anche come funzione Matlab (`nnls`), in ogni caso però, la lentezza di questi metodi compromette la loro applicabilità nell'ambito della fattorizzazione non negativa. Una soluzione alternativa è stata trovata nell'imporre, a posteriori, che tutti gli elementi negativi dei due vettori siano uguali a zero. Questa vincolo non trova una giustificazione teorica ma, in ogni caso, risulta essere una buona alternativa pratica.

L'algoritmo ACLS applicato nell'ambito della fattorizzazione non negativa risulta quindi avere la seguente struttura.

Algoritmo:

1. Dati in input λ_W,λ_H;
2. Inizializzazione di $W \in \mathbb{R}^{m\times k}$;
3. Per j=1,2,... fino a convergenza
   (a) Risolvere, nella variabile H, il sistema di equazioni
   $$(W^TW + \lambda_H I_k)H = W^TA; \tag{4.6}$$
   (b) Imporre che tutti gli elementi negativi di H siano zero;
   (c) Risolvere, nella variabile W, il sistema di equazioni
   $$(HH^T + \lambda_W I_k)W^T = HA^T;$$
   (d) Imporre che tutti gli elementi negativi di W siano zero.

In precedenza con I_k si indica la matrice identità di ordine k.

Il sistema di equazioni (4.6) corrisponde a risolvere i problemi di minimo della forma (4.5) infatti sia

$$F(h) = ||a_j - Wh_j||_2^2 + \lambda_H||h_j||_2^2 = h_j^T(W^TW + \lambda_H I_k)h_j - 2h_j^TWa_j + a_j^Ta_j.$$

Allora, per $F'(h) = 0$ si ha

$$(W^TW + \lambda_H I_k)h_j = W^Ta_j \qquad j = 1, \ldots, n.$$

Si nota come l'algoritmo ACLS usi semplicemente la norma euclidea dei vettori h_j e w_j nella penalizzazione della densità delle matrici H e W. In letteratura sono state proposte strategie alternative: l'algoritmo AHCLS (Alternating Hoyer Constrained Least Squares), per esempio, introduce una parametro più raffinato per l'imposizione del vincolo di sparsità:

$$\text{spar}(x) = \frac{\sqrt{n} - \dfrac{||x||_1}{||x_2||_2}}{\sqrt{n} - 1}, \qquad x \in \mathbb{R}^n.$$

L'implementazione del metodo prevede quindi che siano dati in input, oltre ai parametri λ_W e λ_H, anche due nuovi scalari α_W e α_H tali che $0 \leq \alpha_W, \alpha_H \leq 1$. Questi scalari agiscono sulla sparsità di ogni colonna delle matrici W, H e, essendo compresi tra zero e uno, possono essere facilmente interpretati come la "percentuale" di sparsità che si vuole imporre.
Si nota come fissando α_W e α_H si ha che

$$\alpha_H = \text{spar}(h_j) \quad \Leftrightarrow \quad (\alpha_H - \sqrt{k}(\alpha_H - 1))^2 - \frac{\| h_j \|_1^2}{\| h_j \|_2^2} = 0$$

$$\Leftrightarrow \quad (\alpha_H - \sqrt{k}(\alpha_H - 1))^2 \| h_j \|_2^2 - \| h_j \|_1^2 = 0$$

Ponendo $\beta_H := (\alpha_H - \sqrt{k}(\alpha_H - 1))^2$ si ottiene

$$\beta_H h_j^T h_j - h_j^T \mathbf{1}\mathbf{1}^T h_j = 0,$$

dove con h_j si indica la j-esima colonna della matrice $H \in \mathbb{R}^{k \times n}$ e con $\mathbf{1}$ il vettore colonna composto da vettori tutti uguali a uno: $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^k$. Gli stessi risultati si ottengono con la matrice W.

L'algoritmo avrà dunque la seguente struttura:
Algoritmo:

1. Dati in input λ_W, λ_H, α_W e α_H;
2. Inizializzazione di $W \in \mathbb{R}^{m \times k}$;
3. Calcolo della costante $\beta_W = ((1 - \alpha_W)\sqrt{k} + \alpha_W)^2$;
4. Calcolo della costante $\beta_H = ((1 - \alpha_H)\sqrt{k} + \alpha_H)^2$;
5. per j=1,2,... fino a convergenza
   (a) Risolvere, nella variabile H, il sistema di equazioni
   $$(W^T W + \lambda_H \beta_H I_k - \lambda_H E_k)H = W^T A; \tag{4.7}$$
   (b) Imporre che tutti gli elementi negativi di H siano zero;
   (c) Risolvere, nella variabile W, il sistema di equazioni
   $$(H^T H + \lambda_W \beta_W I_k - \lambda_W E_k)W^T = HA^T;$$
   (d) Imporre che tutti gli elementi negativi di W siano zero.

dove $E_k \in \mathbb{R}^{k \times k}$ è la matrice quadrata composta da tutti elementi uguali a uno.

Le equazioni che compongono i sistemi lineari della forma (4.7) sono equivalenti ai problemi di minimo

$$\forall j \qquad \min_{h_j} ||a_j - Wh_j||_2^2 + \lambda_H(\beta_H h_j^T h_j - h^T \mathbf{1}\mathbf{1}^T h), \qquad \lambda_H \geq 0, h_j \geq 0. \tag{4.8}$$

Infatti sia

$$\begin{aligned} F(h) &= ||a_j - Wh_j||_2^2 + \lambda_H(\beta_H h_j^T h_j - h^T \mathbf{1}\mathbf{1}^T h) = \\ &= h_j^T(W^T W + \lambda_H(\beta_H I_k - \mathbf{1}\mathbf{1}^T h_j - 2h_j^T W a_j + a_j^T a_j, \end{aligned}$$

allora

$$F'(h) = (W^T W + \lambda_H \beta_H I_k - \lambda_H \mathbf{1}\mathbf{1}^T)h_j = W^T a_j, \qquad j = 1, \ldots, n,$$

e per $F'(h) = 0$ si risolve il problema (4.8).

L'algortimo AHCLS quindi assomiglia molto a quello ACLS ma gli esperimenti numerici effettuati in [3] mostrano come questo fornisca una maggiore sparsità delle matrici W e H.

4.3 Considerazioni computazionali

Gli algoritmi presentati godono di numerosi vantaggi. Ad esempio, invece di determinare la matrice H colonna per colonna, si risolve un problema ai minimi quadrati della forma dell'equazione (4.4) calcolando quindi una sola soluzione di un sistema lineare ad ogni iterazione. In più questo sistema lineare è della forma $k \times k$ con $k \ll m, n$ e questo porta ad una grande velocità degli algoritmi ACLS e AHCLS.

Un altro evidente vantaggio è quello che, durante le iterazioni, un elemento di W o di H che si annulla, rimane uguale a zero anche per tutte le iterazioni successive. Questo aspetto è molto utile nell'ambito del *text mining*. Infatti, come è già stato presentato, le colonne di W possono essere interpretate come argomenti, vettori della base dello spazio dei documenti, e questa caratteristica di ACLS e AHCLS fa sì che possano essere solo aggiunti e non eliminati termini, cioè componenti non nulle, alla base. Come risultato, se l'algoritmo si indirizza verso la direzione di un determinato vettore-argomento, le iterazioni continuano a percorre quella direzione. D'altra parte però, i metodi ALS non 'bloccano' gli elementi e questo garantisce una grande flessibilità permettendo di uscire da una direzione che porta ad minimo locale poco significativo.

Per quanto riguarda la convergenza, si dimostra che i metodi ALS convergono ad un punto fisso ma questo può essere un estremante locale o un punto sella. Ad esempio gli algoritmi ACLS e AHCLS in cui il vincolo di non negatività viene imposto con il metodo NNLS, convergono ad un minimo locale.

Per gli algoritmi invece presentati in questa tesi, con un vincolo di non negatività a posteriori che velocizza l'implementazione e ne accresce la sparsità, non è ancora stata data prova della loro convergenza ad un minimo locale (questo problema si riscontra in numerosi algoritmi della NMF).

Capitolo 5

Applicazione al text mining

Come è già stato anticipato in questa tesi, con il termine *text mining* si indicano tutti quei metodi usati per estrarre informazioni utili da una grande e spesso non strutturata collezione di testi.
Per applicare i metodi presentati in precedenza, si utilizzano le informazioni contenute all'interno del database MEDLINE[1] mediante le matrici `A_med` e `dict_med` visibili nel sito:

`http://www.nlm.nih.gov/bsd/pmresources.html`

Questo database contiene un enorme numero di articoli scientifici di argomento medico, alcuni dei quali sono rappresentati dalla matrice `A_med`. Questa matrice, che indicheremo con A, definisce quindi la matrice *termini*$\times$*documenti*. Infatti la matrice A è strutturata in modo da avere sulle colonne i documenti che, in questo caso, sono appunto articoli di stampo medico, mentre, sulle righe, i termini chiave, quelli che sono più frequenti all'interno della collezione dei documenti. Così, l'elemento $a_{ij} \in A$ rappresenta il numero di occorrenze con cui il termine i-esimo appare nel j-esimo documento.

[1] Gli articoli contenuti all'interno di MEDLINE sono stati organizzati nella matrice `A_med` dai gestori del sito `http://scgroup20.ceid.upatras.gr:8000/tmg/`.

La matrice A è molto ampia, $A \in \mathbb{R}^{5735\times1033}$, non negativa e sparsa.

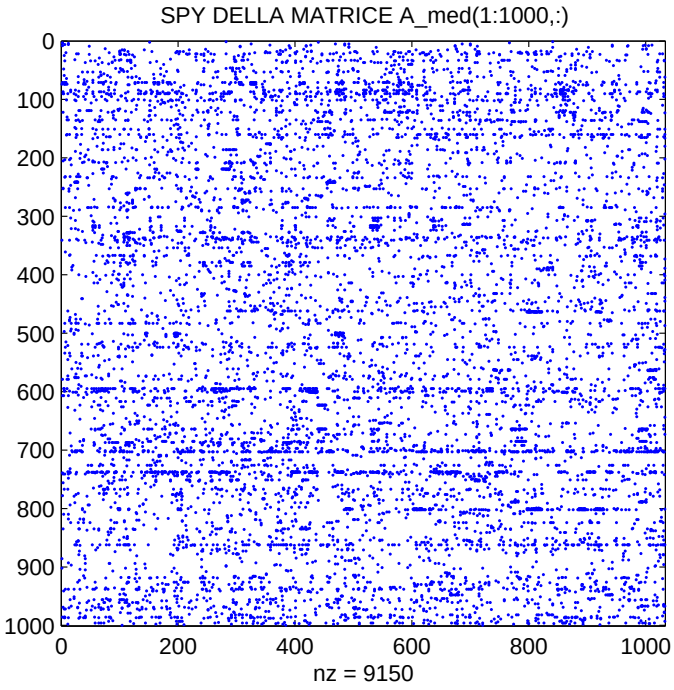

Figura 5.1: I puntini indicano gli elementi non nulli della porzione di `A_med`

In Figura 5.1 è rappresentato il pattern di sparsità di una porzione della matrice A di dimensioni 1000×1033.

La matrice `dict_med`, di dimensione 5735×23, invece, rappresenta un "dizionaro" con il quale interpretare i risultati ottenuti dall'applicazione dei metodi.

5.1 LSI: Latent Semantic Indexing

La Latent Semantic Indexing (LSI) si basa sull'assunzione che ci sia qualche struttura latente nei dati, come termini particolarmente ricorrenti, possibilità di raggruppare i documenti per argomento, ecc, e che questa struttura semantica possa essere rilevata proiettando la matrice *termini*$\times$*documenti* in uno spazio di dimensione minore tramite la SVD troncata. Se A è la matrice *termini*$\times$*documenti*, allora, applicandole la SVD troncata, se ne genera una approssimazione di rango k:

$$A \approx A_k = U_k \Sigma_k V_k^T =: U_k H_k, \tag{5.1}$$

Riscrivendo la relazione (5.1) colonna per colonna, si ha

$$a_j \approx U_k h_j.$$

Ciò significa che la j-esima colonna di H_k, h_j, contiene le coordinate del j-esimo documento in termini della base ortogonale U_k.

Per A di grandi dimensioni, come nel nostro caso, questa fattorizzazione viene fatta applicando il metodo di Lanczos. Infatti questo metodo, come è già stato presentato, permette di scegliere a priori la dimensione k dello spazio approssimante evitando quindi operazioni inutili e rendendo il metodo più veloce. Si nota come le colonne di U_k appartengano allo spazio dei documenti e ne costituiscano una base approssimante ortogonale. La proprietà di ortogonalità della base approssimante è molto utile, infatti, interpretando i vettori della base come vettori "argomento", questo permette di avere argomenti "ortogonali" cioè disgiunti gli uni dagli altri.

Un metodo per determinare, in prima approssimazione, una buona dimensione k dello spazio approssimante è analizzare l'andamento dei valori singolari di A. Nell'ambito del *text mining* questi hanno spesso un andamento a gomito e quindi, i valori singolari scelti per l'approssimazione saranno i primi k per cui questo andamento decresce rapidamente.

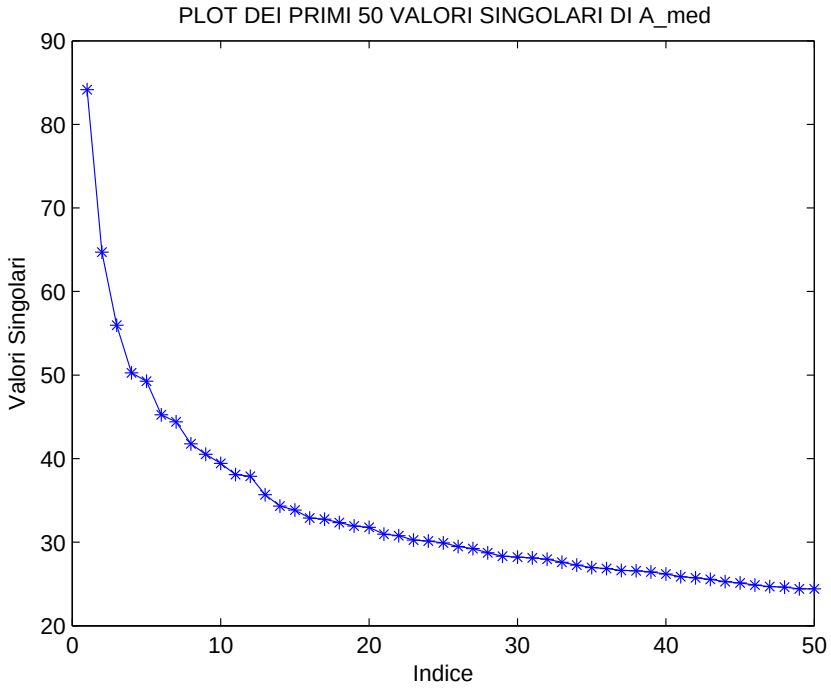

Figura 5.2: Andamento dei primi 50 valori singolari di `A_med`

Nel caso descritto in Figura 5.2, la dimensione ottimale dello spazio approssimante è tra dieci e venti.

Un grande svantaggio della LSI però, dovuto all'applicazione della SVD, è la perdita della struttura di non negatività e di sparsità della matrice `A_med`. Infatti la base ortogonale U_k è densa e contiene anche elementi negativi. Questo porta ad una difficile interpretazione rispetto al dizionario `dict_med`.

5.2 Applicazione della NMF

Nell'ambito di questa tesi, per l'applicazione della fattorizzazione non negativa, vengono usati gli algoritmi ACLS e AHCLS presentati in precedenza. Una volta fattorizzata la matrice *termini*×*documenti* A, cioè

$$A \approx WH, \qquad W \geq 0, \quad H \geq 0,$$

dove $W \in \mathbb{R}^{m\times k}$ e $H \in \mathbb{R}^{k\times n}$, si ha che la j-esima colonna di H contiene le coordinate del j-esimo documento nei termini della base approssimante e non ortogonale composta dalle colonne di W: $\{w_1, \ldots, w_k\}$. La peculiarità della fattorizzazione non negativa è quella di fornire vettori di base a componenti non negative e quindi facilmente interpretabili in termini del dizionario `dict_med`. Andando a considerare le componenti maggiori del j-esimo vettore di base w_j e interpretandole, si può assegnare una "etichetta" (argomento) a w_j. Questi "argomenti" però, non saranno totalmente disgiunti gli uni dagli altri, come nella LSI, poichè la base $\{w_1, \ldots, w_k\}$ non è ortogonale. Presentiamo le 15 componenti maggiori dei primi 5 vettori di base dello spazio approssimante di dimensione $k = 25$ ottenuti applicando l'algoritmo ACLS con parametri $\lambda_H = \lambda_W = 0.5$.

w_1 : $[glucose, fatty, acids, insulin, acid, ffa, free, serum, blood, infusion, levels, maternal, increased, values, metabolism]$;

w_2 : $[body, image, syndrome, verbal, understanding, impairment, disturbances, speech, gerstmann, lesion, aphasic, case, disease, aphasia, words]$;

w_3 : $[alveolar, inclusion, bodies, lung, pulmonary, surface, epithelial, respiratory, membrane, lining, lungs, air, electron, normal, layer]$;

w_4 : $[aortic, left, ventricular, percent, regurgitation, valve, flow, regurgitant, patients, stroke, volume, total, replacement, ventricle, normal]$;

w_5 : $[flow, coronary, hypothermia, perfusion, pressure, dogs, bypass, cooling, rate, total, temperature, induction, saturation, results, reduced]$.

Quindi ad esempio al primo vettore di base w_1 si può assegnare l'etichetta **"sangue e glicemia"** mentre al terzo w_3, **"apparato respiratorio"**.

Proponiamo anche le 15 componenti maggiori dei primi 5 vettori di base ottenuti sempre con una approssimazione di ordine $k = 25$ ma applicando l'algoritmo AHCLS con parametri $\lambda_H = \lambda_W = 0.5$ e $\alpha_H = \alpha_W = 0.9$.

w_1' : $[oxygen, tension, cerebral, arterial, pressure, blood, cisternal, fluid, rats, dioxide, carbon, po2, increased, increase, air]$;

w_2' : $[hgh, serum, growth, human, hormone, hypopituitary, rate, patients, treatment, nitrogen, withdrawal, antibodies, period, anti, children]$;

w_3' : $[protein, soluble, cataract, lens, fractions, lenses, processes, slow, development, insoluble, increase, quantity, fraction, disappearance, investigations]$;

, w_4' : $[liver, hepatitis, type, cells, bile, giant, biliary, histological, clinical, infants, neonatal, stasis, cell, ducts, findings]$;

w_5' : $[glucose, fatty, acids, insulin, acid, blood, increased, ffa, free, serum, levels, plasma, level, maternal, infusion]$.

Quindi, nonostante la parità di parametri, l'algortimo AHCLS produce una base differente in cui il vettore w_5', ad esempio, è molto simile al vettore w_1 prodotto dall'algoritmo ACLS, e si può etichettare quindi come **"sangue e glicemia"**.

Un'altra possibilità data dalla NMF è quella di studiare lo stesso database di documenti, cioè la struttura della matrice `A_med`, studiando in particolare la matrice H. Infatti, come è già stato detto, la j-esima colonna h_j di H contiene le componenti del j-esimo documento del database in termini della base approssimante $\{w_1, \ldots, w_k\}$. Ad esempio, proponiamo una porzione 5×7 di H generata dall'algoritmo ACLS applicato con un ordine di approssimazione $k = 25$ con parametri $\lambda_H = \lambda_W = 0.5$.

$$H(1:5, 1:7) = \begin{pmatrix} 2.1219 & 0.2317 & 0.0869 & 0.6716 & 1.9530 & 1.8502 & 1.1103 \\ 0 & 0 & 0 & 0 & 0.7729 & 0.2605 & 0 \\ 0 & 0.1142 & 0.7309 & 0.0857 & 0.0207 & 0 & 0.3080 \\ 0.0280 & 0 & 0 & 0.1491 & 0 & 0 & 0.4199 \\ 0 & 0 & 0.0635 & 0 & 0 & 0 & 0.0716 \end{pmatrix}.$$

Si nota come la prima componente del vettore h_1, cioè la prima colonna di H, sia molto grande. Questo significa che scrivendo h_1 in termini della base $\{w_1, \ldots, w_{25}\}$ si ha una grande componente lungo w_1 e che quindi il primo documento del database rappresentato dalla matrice `A_med` parla di **"sangue e glicemia"**, l'etichetta assegnata a w_1.

Proponiamo ora la stessa porzione della matrice H ottenuta però con l'algoritmo AHCLS con un ordine di approssimazione $k = 25$ e parametri $\lambda_H = 0.5$, $\lambda_W = 0.5$ e $\alpha_H = \alpha_W = 0.9$.

$$H'(1:5, 1:7) = \begin{pmatrix} 0 & 0 & 0.3084 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0482 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.6762 & 0.1975 & 0.1203 & 0.6823 & 1.6115 & 1.5980 & 0.9046 \end{pmatrix}.$$

Si nota subito come l'algoritmo AHCLS fornisca una maggiore sparsità rispetto all'algoritmo ACLS e che permette quindi una più facile interpretazione. Considerando il vettore h'_1, prima colonna di H', si nota come abbia una grande componente in termini del quinto vettore della base approssimante w'_5 a cui è stata comunque assegnata l'etichetta **"sangue e glicemia"**. Questo permette di affermare con ancora più sicurezza che il primo documento del database considerato tratta questa tematica.

5.3 Confronto computazionale

Per quanto riguarda la SVD, si può calcolare il residuo relativo fornito dal metodo come

$$r_{svd}(k) = \frac{||A - A_k||_2}{||A||_2}.$$

Questa grandezza è da interpretare come la distanza tra la matrice A e la sua approssimazione A_k. Analogamente, si calcola il residuo fornito dalla NMF come

$$r_{nmf}(k) = \frac{||A - W_k H_k||_F}{||A||_F}.$$

Si nota che, in questa applicazione, il residuo può essere visto come funzione di k, ordine dell'approssimazione. Si vuole che questa funzione sia decrescente così che, all'aumentare dell'ordine di approssimazione, la distanza tra A e la sua approssimazione, A_k per la SVD e $W_k H_k$ per la NMF, diminuisca. Questo accade sicuramente per il residuo ottenuto con la SVD ma non si può dire la stessa cosa per quanto riguarda la NMF. Infatti, come è già stato detto, non è ancora stato provato che la NMF, applicata con algoritmi ACLS e AHCLS, converga ad un minimo assoluto. Può quindi accadere che la NMF, calcolata con un ordine di approssimazione k, converga ad un minimo relativo che fornisce però un residuo maggiore rispetto a quello fornito dal minimo relativo raggiunto con un ordine di approssimazione $l < k$.

Come è già stato presentato in questa tesi, la SVD fornisce l'approssimazione più accurata in termini di residuo.

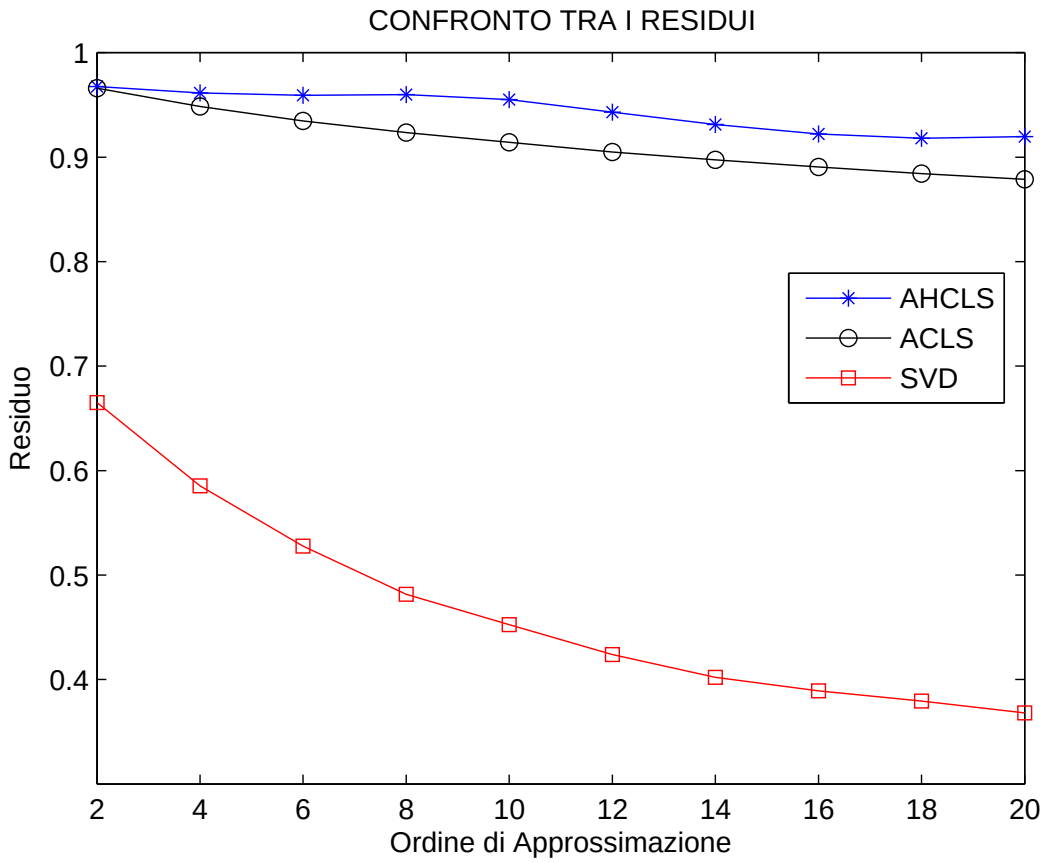

Figura 5.3: Residui ottenuti con SVD, e NMF con algoritmi ACLS e AHCLS

Si nota come il residuo della NMF, applicata con l'algoritmo AHCLS, non sia decrescente, ma, ad esempio, $r(8) > r(6)$.

D'altra parte però abbiamo mostrato come la NMF fornisca un gran numero di possibilità interpretative in più rispetta alla SVD usata nell'ambito della LSI per il *text mining*. Per rendere i dati forniti dalla NMF affidabili però, il metodo va applicato più volte cambiando anche il punto iniziale cioè l'inizializzazione di W. Questo perchè la NMF è un metodo iterativo e quindi dipende in maniera significativa dal punto iniziale e, in più, come è già stato detto, non è ancora stato dimostrato che la NMF applicata con algoritmo AHCLS converga ad un minimo globale.

Per i risultati presentati fino a questo punto si è inizializzata la matrice W con delle componenti casuali comprese tra 0 e 1.

Una inizializzazione alternativa può essere la seguente:

1. Data A matrice non negativa, si fattorizza tramite la SVD troncata di ordine k:

$$A \approx U_k \Sigma_k V_k^T$$

con $U_k = [u_1, \ldots, u_k]$ e $V_k = [v_1, \ldots, v_k]$;

2. Essendo A non negativa, anche u_1 lo è, allora si pone $W(:,1) = u_1$;

3. u_2 ha sicuramente componenti non negative, allora si calcola $C^{(2)} = u_2 v_2^T$;

4. Si pongono tutti gli elementi negativi di $C^{(2)}$ uguali a zero;

5. Si pone

$$W(:,2) = u_1 C_+^{(2)}$$

dove con $C_+^{(2)}$ si indica la matrice $C^{(2)}$ in cui gli elementi negativi sono stati posti uguale a zero;

6. Si riparte dal punto 2. fino a determinare $W(:,k)$.

Con questo nuovo tipo di inizializzazione si ottengo nuovi vettori della base approssimante. Presentiamo le 10 componenti maggiori dei primi 5 vettori di base ottenuti con un'approssimazione di ordine $k = 25$ e algoritmo AHCLS di parametri $\lambda_H = \lambda_W = 0.5$ e $\alpha_H = \alpha_W = 0.9$ in cui la matrice W è stata inizializzata nel modo appena mostrato:

$$\begin{aligned}
w_1 :&\ [growth, hormone, human, hgh, treatment, effect, rats,\\
&\ serum, pituitary, patients];\\
w_2 :&\ [cells, marrow, cell, lymphocytes, bone, hela, small,\\
&\ thymidine, thymus, blood];\\
w_3 :&\ [growth, hormone, human, hgh, treatment, effect, rats,\\
&\ serum, pituitary, tumor];\\
w_4 :&\ [dna, kidney, phage, subtilis, weight, nephrectomy,\\
&\ rna, synthesis, days, acid];\\
w_5 :&\ [cases, ventricular, septal, defect, left, type, aortic,\\
&\ pulmonary, pressure, cardiac].
\end{aligned}$$

Analizzando queste componenti si può, ad esempio, assegnare l'etichetta **"midollo osseo"** al vettore di base w_2.

I risultati ottenuti con questa inizializzazione, permettono di notare, ancora una volta, come la NMF fornisca una base approssimante non ortogonale: le prime 7 componenti principali dei vettori w_1 e w_3 sono uguali e quindi i due vettori hanno direzioni molto simili. Questo significa che gli argomenti individuati dai vettori di base non sono disgiunti gli uni dagli altri.

Bibliografia

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

[2] Daniel D. Lee and H. Sebastian Seung, *Algorithms for Non-negative Matrix Factorization*, Advances in Neural Information Processing Systems 13, (2001), 556-562.

[3] Russell Albright, James Cox, David Duling, Amy N. Langville, and Carl D. Meyer, *Algorithms, Initializations, and Convergence for the Nonnegative Matrix Factorization*, NCSU Technical Report Math 81706.

[4] Abraham Berman, Robert J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM.

[5] Charles L. Lawson, Richard J. Hanson, *Solving Least Squares Problems*, SIAM, 1995.

[6] Gene H. Golub, Charles F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1990.