PLS/LBD vs. PCR - Un analisi mediante il fattore di rimpicciolimento

Barbarino Giovanni

31 marzo 2014

Sommario

In questa relazione del lavoro di *Lars Eldén, 2002, Partial least-squares vs. Lanczons Bidiagonalization–I: analysis of a projection method for multiple regression,* analizzeremo gli algoritmi Partial Least-Squares(PLS), Lanczos Bidiagonalization(LBD), ne mostreremo l'equivalenza, e li confronteremo con l'algoritmo Principal Components Regression(PCR), mediante l'utilizzo del fattore di rimpicciolimento (Shrinkage Factor).

Indice

1 Introduzione

Consideriamo il problema ai minimi quadrati

$$\min_{x\in\mathbb{R}^n} \|Ax - y\|$$

dove A è una matrice reale $m\times n$, e y è un vettore di $\mathbb{R}^m$ noto. Spesso, a causa del cattivo condizionamento della matrice, o delle sue eccessive dimensioni, conviene studiare le soluzioni al problema ristretto ad un particolare sottospazio di $\mathbb{R}^n$. Chiamata V una matrice $m\times k$, che abbia come colonne una base del sottospazio scelto, si cerca quindi di risolvere

$$\min_{x\in\mathbb{R}^k} \|AVx - y\| \tag{1}$$

Uno dei metodi usati per proiettare le soluzioni su un sottospazio di dimensione piccola, è il *principal component regression (PCR)*, noto in algebra lineare come *decomposizione ai valori singolari troncata (TSVD)*, che tratteremo nel Capitolo 2. Altri due metodi usati nel risolvere questo problema, e che già da tempo sappiamo essere equivalenti (Wold, 1984), sono il *partial least-squares (PLS)* e la *Lanczos Bidiagonalization (LBD)*, che mostreremo nel Capitolo 3.

Il PLS è un metodo le cui proprietà non sono state ancora completamente analizzate, anche a causa del fatto che *la soluzione non è una funzione lineare dei dati*, ma nonostante ciò, è un metodo ampiamente utilizzato in chemiometria, in quanto spesso porta ad una riduzione del residuo più velocemente rispetto al PCR tradizionale. Negli ultimi capitoli, quindi, confronteremo gli algoritmi tramite esempi numerici, e ne analizzeremo la velocità di convergenza tramite l'utilizzo del *coefficiente di rimpicciolimento (Shrinkage Factor)*, che mostra come i valori singolari siano essenziali nel comprendimento delle proprietà del PLS.

Infine, daremo spunti di possibili ulteriori sperimentazioni effettuabili sugli algoritmi, riguardanti la loro efficienza e stabilità.

2 PCR

Nei casi in cui la matrice A del problema ai minimi quadrati sia quasi singolare, la soluzione puó venire di norma elevata. L'idea del PCR é di scrivere la soluzione solo in relazione a prime poce componenti principali.

Supponiamo ora di scrivere la matrice nella sua scomposizione SVD

$$A = S\begin{pmatrix}\Sigma_r & 0\\ 0 & 0\end{pmatrix}Q^t, \qquad \Sigma_r = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$$
$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$$

dove S e Q sono matrici ortogonali, rispettivamente $m\times m$ e $n\times n$, con colonne s_i e q_i chiamati vettori singolari (o componenti principali), e r è il rango della

matrice A. Assumendo $m \geq n \geq r$, allora la soluzione al problema ai minimi quadrati è espressa da

$$x = A^{+}y = \sum_{i=1}^{r} \frac{s_i^t y}{\sigma_i} q_i$$

Applicare il PCR equivale a troncare la soluzione data dall'SVD ai primi k termini, proiettando la soluzione sul sottospazio generato dalle prime k colonne di Q. chiamata $x_{pcr}^{(k)}$ la soluzione data dal PCR,

$$Q_k = (q_1 q_2 \dots q_k), \qquad \left\| Ax_{pcr}^{(k)} - y \right\| = \|AQ_k z_k - y\| = \min_{x \in \mathbb{R}^k} \|AQ_k x - y\| \quad (2)$$

$$\implies x_{pcr}^{(k)} = \sum_{i=1}^{k} \frac{s_i^t y}{\sigma_i} q_i, \qquad z_k = \left(\frac{s_1^t y}{\sigma_1}, \frac{s_2^t y}{\sigma_2}, \dots, \frac{s_k^t y}{\sigma_k} \right)^t \quad (3)$$

É possibile però, tramite semplici calcoli, valutare il residuo associato a tale soluzione.

$$R_k^2 = \left\| Ax_{pcr}^{(k)} - y \right\|^2 = \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} z_k - S^t y \right\|^2 = \sum_{i=k+1}^{m} (s_i^t y)^2 \quad (4)$$

Il programma *pcr.m* allegato calcola la soluzione $x_{pcr}^{(k)}$ e il residuo secondo le formule (2) e (3), mentre *pcr_ esatto.m* calcola il residuo quadro con la formula (4). Questa mostra che la scelta del PCR non è la migliore possibile: per ridurre il residuo più velocemente, si dovrebbero prendere invece le componenti nell'ordine indicato dalla grandezza delle componenti di $S^t y$. Il fatto che il PCR non tenga in considerazione il vettore noto, ma lavori solo sulla matrice, compromette dunque la sua velocità di convergenza, cosa che invece, come vedremo, non accade nei metodi PLS e LBD.

3 PLS e LBD

3.1 NIPALS PLS

Diamo una descrizione, in pseudo-codice, dell'algoritmo in *pls.m*

$A_0 = A$
for $i = 1, 2, \dots, k$ **do**
$w_i = A_{i-1}^t y \qquad w_i = w_i / \|w_i\|$
$t_i = A_{i-1} w_i \qquad t_i = t_i / \|t_i\|$
$p_i = A_{i-1}^t t_i$
$A_i = A_{i-1} - t_i p_i^t$
end for

Per studiarne le proprietà, definiano $K_i(A, y) = span\{y, Ay, \ldots . A^{i-1}y\}$ il Sottospazio di Krilov generato dalla matrice A e dal vettore y. Vale il seguente risultato:

Teorema 1. *I vettori $w_1, w_2, \ldots, w_i$ sono ortonormali, e generano $K_i(A^tA, A^ty)$ Inoltre, i vettori $t_1, t_2, \ldots, t_i$ sono ortonormali, e generano $K_i(AA^t, AA^ty)$*

Definendo ora W_k, T_k, P_k le matrici che hanno i vettori w, t, p come colonne, la soluzione determinata dall'algoritmo al problema dei minimi quadrati è

$$x_{pls}^{(k)} = W_k(P_k^tW_k)^{-1}T_k^ty \tag{5}$$

dal Teorema 1, e da (5), concludiamo che il PLS trova la soluzione ad (1) sul sottospazio $K_i(A^tA, A^ty)$. Il programma *pls.m* calcola la soluzione del PLS usando (5), e ne ricava il residuo.

3.2 LBD

L'LBD si propone di ridurre A ad una matrice bidiagonale triangolare superiore B_k, tramite due matrici ortogonali V_k e U_k. Una descrizione dell'algoritmo usato in *lbd.m*, in pseudo-codice, è riportata di seguito:

$v_1 = A^ty \qquad v_1 = 1_1/\|v_1\|$
$u_1 = Av_1 \qquad \alpha_1 = 1/\|u_1\| \qquad u_1 = \alpha_1 u_1$
for $i = 2, 3, \ldots, k$ **do**
$\qquad v_i = A^tu_{i-1} - \alpha_{i-1}v_{i-1} \qquad \gamma_{i-1} = 1/\|v_i\| \qquad v_i = \gamma_{i-1}v_i$
$\qquad u_i = Av_i - \gamma_{i-1}u_{i-1} \qquad \alpha_i = 1/\|u_i\| \qquad u_i = \alpha_i u_i$
end for

Se V_k e U_k hanno per colonne i primi k vettori u e v, e B_k è la matrice

$$B_k = \begin{pmatrix} \alpha_1 & \gamma_1 & & & & \\ & \alpha_2 & \gamma_2 & & & \\ & & \ddots & \ddots & & \\ & & & & \alpha_{k-1} & \gamma_{k-1} \\ & & & & & \alpha_k \end{pmatrix}$$

allora avremo $AV_k = U_kB_k$.

È possibile dimostrare il seguente risultato

Teorema 2. *I vettori $v_1, v_2, \ldots, v_i$ sono ortonormali, e generano $K_i(A^tA, A^ty)$. Inoltre, i vettori $u_1, u_2, \ldots, u_i$ sono ortonormali, e generano $K_i(AA^t, AA^ty)$*

Risolviamo ora (1) ristretto al sottospazio $K_i(A^tA, A^ty)$:

$$\begin{aligned}\min_z \|AV_k z - y\|^2 &= \min_z \|U_k B_k z - y\|^2 \\ &= \min_z \left\| \begin{pmatrix} U_k^t \\ U_\perp^t \end{pmatrix} (U_k B_k z - y) \right\|^2 \\ &= \min_z \|B_k z - U_k^t y\|^2 + \|U_\perp^t y\|^2\end{aligned}$$

La soluzione, quindi, diventa

$$z = B_k^{-1} U_k^t y \implies x_{lbd}^{(k)} = V_k B_k^{-1} U_k^t y$$

Il programma *lbd.m* usa questa formula per calcolare la soluzione data dall'LBD, e trovare il residuo.

3.3 Equivalenza degli algoritmi

Dai Teoremi 1 e 2, possiamo vedere che entrambi gli algoritmi trovano la soluzione ottimale allo stesso sottospazio di Krilov $K_i(A^tA, A^ty)$, e notando che $v_1 = w_1$ e $u_1 = t_1$, allora per induzione si prova un risultato più forte:

Teorema 3. *I metodi PLS e LBD generano la stessa base ortonormale (a meno di segno) dello stesso spazio di Krilov, e le due soluzioni sono uguali* $x_{pls}^{(k)} = x_{lbd}^{(k)}$

I due algoritmi, da un punto di vista algebrico, sono dunque equivalenti, ma dal punto di vista computazionale, hanno diverse differenze.

Per esempio, in caso A sia di rango basso, o strutturata, o sparsa, il PLS distrugge tali proprietà variando la matrice mediante l'operazione $A_i = A_{i-1} - t_i p_i^t$. Inoltre, per trovare la soluzione del PLS, abbiamo bisogno di calcolare $P_k W_k$, cosa non necessaria nell'LBD, in quanto B_k è immediatamente disponibile.

Altri possibili confronti tra i due algoritmi saranno considerati nell'ultimo capitolo, ma d'ora in poi indicheremo con $x^{(k)}$ la soluzione al problema ristretto al sottospazio di Krilov $K_k(A^tA, A^ty)$, data da entrambi gli algoritmi.

3.4 Dipendenza della soluzione dai dati

Dire $x^{(k)} \in K_k(A^tA, A^ty)$, è equivalente ad affermare l'esistenza di un polinomio $p_{k-1}(x)$ di grado $k-1$ tale che $x^{(k)} = p_{k-1}(A^tA)A^ty$.

Sia $A = SCQ^t$ la scomposizione SVD di A, con C quadrata diagonale, e S, Q ortogonali rettangolari. Allora possiamo scrivere la norma del residuo come

$$\|Ax^{(k)} - y\|^2 = \|SCp_{k-1}(C^2)CS^ty - y\|^2 = \|(p_k(C^2) - I)S^ty\|^2$$

dove $p_k(x) = xp_{k-1}$.

$$\|Ax^{(k)} - y\|^2 = \sum_{i=1}^{r} (p_k(\sigma_i^2) - 1)^2 + \sum_{i=r+1}^{n} (s_i^t y)^2$$

Questo mostra che la soluzione al problema è equivalente a risolvere

$$\min_{p_k \in \prod_k} \sum_{i=1}^{r} (p_k(\sigma_i^2) - 1)^2$$

dove $\prod_k$ sono i polinomi di grado k o minore, senza termine noto.

Ciò ci dice che, se k è piccolo, la soluzione dipende in maniera polinomiale dai valori singolari, e se perturbiamo y otteniamo un diverso sottospazio di Krilov, quindi *la soluzione dipende dai dati in maniera non lineare*.

Nel prossimo capitolo, mostreremo che proprio questa dipendenza da y, rende l'algoritmo più efficiente rispetto al PCR, in alcune particolari configurazioni dei dati iniziali.

4 Confronto Sperimentale

In questo capitolo mostriamo esplicitamente degli esempi dell'efficienza del PLS in particolari situazioni, utilizzando il programma *pls.m* descritto sopra.

Innanzitutto, dato che tramite SVD possiamo sempre portare la matrice in forma diagonale a meno di matrici ortogonali, prendiamo A diagonale a valori reali nonnegativi. Inoltre, dato che ci interessano solo i risultati per pochi passi dell'algoritmo, A sarà per semplicità quadrata.

Come mostrato dalla Figura 1, consideriamo una matrice 50×50, in cui i pesi sono concentrati sui primi 20 valori singolari, in maniera uniformemente decrescente, e distinguiamo gli esperimenti a seconda del vettore noto y. Dato che la matrice S dell'SVD è l'identità, studiare $s_i^t y$ al variare di i equivale a studiare le componenti di y, quindi il residuo dipenderà esplicitamente dagli elementi del vettore y.

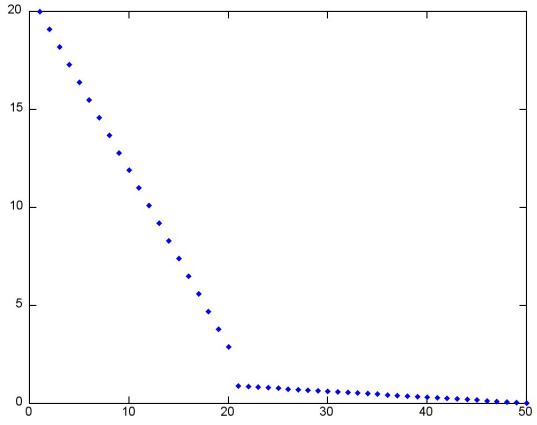

Figura 1: Elementi sulla diagonale di A

4.1 Pesi Concentrati su Poche Componenti

In questa sezione, consideriamo dei termini noti concentrati su tre componenti, per vederne la dipendenza dalle componenti principali della matrice.

Come mostrato in Figura 2, costruiamo due diverse y in modo che la seconda abbia le componenti principali spostate verso i valori singolari più bassi. In Figura 3 vengono riportati i residui prodotti dall'algoritmo PLS nei primi passi. Il comportamento dei sottospazi di Krylov *in questi due esempi*, può essere riassunto come segue:

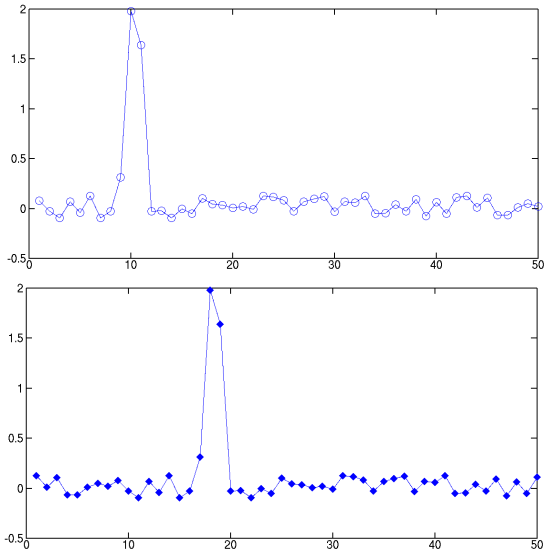

Figura 2: Componenti di y, Esempio 1 (sopra) e 2 (sotto)

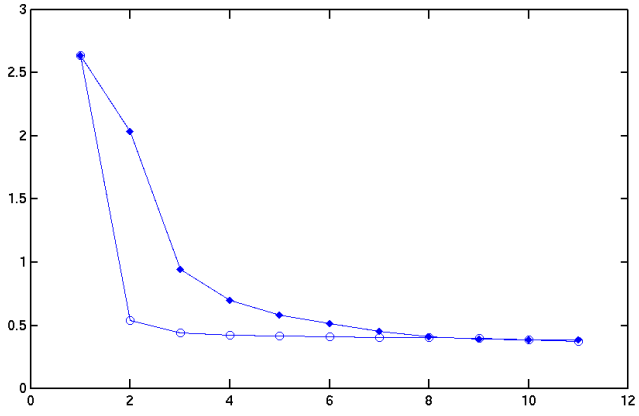

Figura 3: Residuo con il metodo PLS/LBD in funzione del numero di passi

- Se il vettore noto è concentrato lungo dei valori singolari che non corrispondono ai più grandi, ma la massa del vettore è abbastanza grande, allora pochi primi elementi della base di Krylov avranno grandi componenti lungo quei vettori singolari
- Se la massa del vettore non è abbastanza grande in relazione ai valori singolari, allora i primi vettori di base saranno dominati dalle componenti associate ai valori singolari più alti.
- I successivi vettori di base, avranno le componenti più alte lungo i valori singolari predominanti
- Gradualmente, l'influenza dei valori singolari più piccoli si farà sentire, ma a meno che y non abbia componenti consistenti lungo quei valori, ci vorrà molto tempo prima che contribuiscano in modo significativo alla soluzione.

Questi esempi non danno tuttavia il quadro complessivo, in quanto il comportamento del metodo dei sottospazi di Krylov è decisamente più complicato.

4.2 Due Casi Estremi

Diamo altri due esempi di comportamento di questo metodo.

Consideriamo, come Esempio 3, la stessa matrice di sopra, ed un vettore noto con componenti circa uguali. Invece, come Esempio 4, modifichiamo la matrice A ponendo i primi 25 valori singolari tra 1.1 e 1, e gli ultimi 25 a 10^{-3}; prendiamo inoltre, come termine noto, un vettore le cui prime 25 componenti siano uguali e alte, mentre le ultime 25 siano quasi nulle.

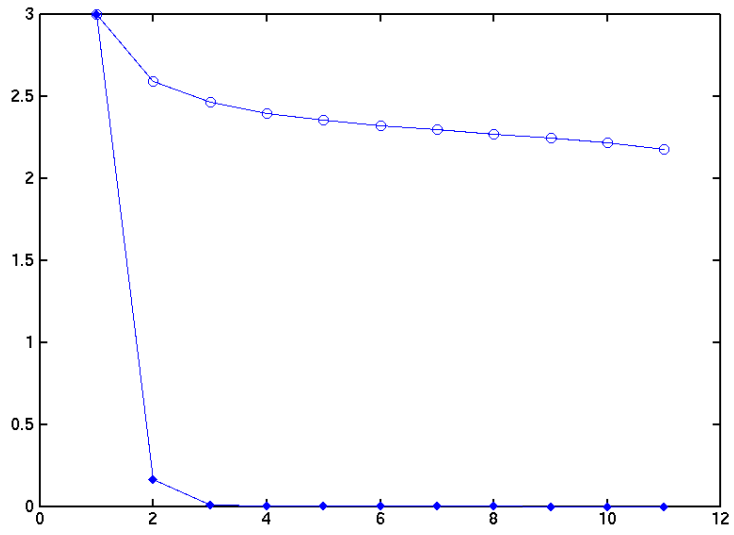

Figura 4: Residuo con il metodo PLS degli Esempi 3 (cerchi) e 4 (rombi)

L'esempio 3 mostra, in linea con le nostre osservazioni, che il decadimento del residuo è molto lento in caso y abbia componenti significative lungo valori singolari piccoli, mentre l'Esempio 4 ci mostra che il PLS tiene in considerazione tutti i valori singolari vicini al massimo contemporaneamente, valorizzando le componenti del termine noto lungo di esse più velocemente del normale.

Altri spunti di esperimenti verranno dati nell'ultimo capitolo, ma ora è il tempo di introdurre un nuovo strumento che ci permetterà un analisi più profonda delle dinamiche degli algoritmi: il fattore di rimpicciolimento.

5 Shrinkage Factor

Come abbiamo visto nel Capitolo 2, il PCR ha come soluzione una combinazione lineare dei vettori q_i, che formano una base ortonormale di $\mathbb{R}^n$. Ciò vuol dire che anche la soluzione generata dal PLS, e in generale qualsiasi vettore, si può scrivere come combinazione dei vettori q_i.

Lo *Shrinkage Factor*, o fattore di rimpicciolimento, è un vettore di lunghezza n, definito in relazione ad un vettore x qualsiasi. In particolare $f(x)$ è il fattore di rimpicciolimento associato a x, se

$$x = \sum_{i=1}^{n} f_i(x) \frac{s_i^t y}{\sigma_i} q_i \tag{6}$$

In aritmetica di macchina, è raro che $s_i^t y$ o σ_i siano esattamente zero, quindi $f(x)$ diventerà molto grande o molto piccolo a seconda dei casi.

Un esempio di come funziona il fattore di rimpicciolimento è dato proprio dal PCR. Da (3), infatti avremo che

$$x_{pcr}^{(k)} = \sum_{i=1}^{k} \frac{s_i^t y}{\sigma_i} q_i \implies f(x_{pcr}^{(k)})_i = \begin{cases} 1 & 1 \le i \le k \\ 0 & k+1 \le i \le r \end{cases}$$

Ovviamente, la soluzione al problema ai minimi quadrati avrà tutte le componenti di f pari a 1, quindi la vicinanza delle componenti di f ad 1, è un indice dalla bontà dell'approssimazione considerata alla soluzione effettiva.

Nel programma *shrink.m*, utilizziamo l'ortogonalità dei q_i per calcolare f in maniera semplice: dato x, avremo che

$$\frac{\sigma_i}{s_i^t y} x^t q_i = f_i(x) \quad \forall i$$

Calcoliamo quindi questo indice per il PLS: in Figura 4 sono riportati i valori del fattore relativi agli esempi 1 e 2, per i primi 5 passi dell'algoritmo.

Come vediamo nell'Esempio 1, il PLS riconosce immediatamente che la massa è concentrata attorno alla decima componente, ponendo il corrispettivo fattore ad 1, e questo valore cambierà di poco nei passaggi successivi. Gradatamente, l'algoritmo incomincia ad uniformare anche le altre componenti ad 1, con velocità proporzionale alla loro massa, e ai valori singolari corrispondenti.

Nell'esempio 2, invece, l'algoritmo impiega due passaggi ad accorgersi che la massa è concentrata verso la diciottesima componente, in quanto influenzato dai bassi valori singolari.

Il fatto che il fattore si mantenga basso in corrispondenza dei valori singolari piccoli, inoltre, è dovuto alle proprietà di stabilità dell'algoritmo: nell'espressione (6), si vede che gli f_i in questione devono mantenersi bassi per contrastare $1/\sigma_i$, e facendo ciò, contribuiscono a mantenere una norma del vettore risultante bassa.

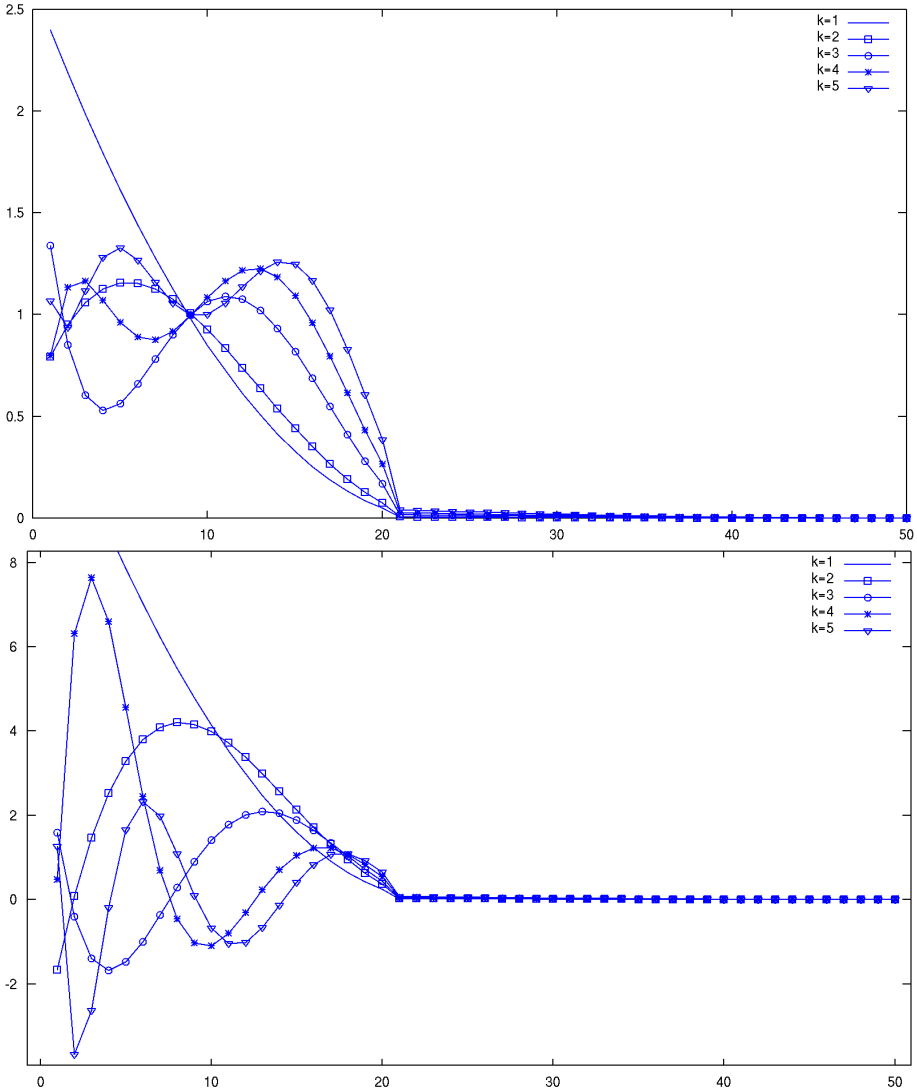

Figura 5: Shrinkage Factor del metodo PLS ai passi 1,2,3,4,5, degli Esempi 1 (sopra) e 2 (sotto)

6 Conclusioni

Nella nostra analisi con i vettori di Krylov, i vettori singolari, e il fattore di rimpicciolimento, abbiamo mostrato che nei primi passi dell'algoritmo PLS, i problemi dovuti alla collinearità dei vettori nella matrice non influenzano di molto la soluzione, nonostante il residuo venga comunque ridotto in maniera significativa. In ogni caso, è necessario un valido metodo di stop, che ci dica quanto possiamo andare avanti nell'algoritmo, prima che errori di approssimazione o altro compromettano seriamente il risultato.

A dispetto di questo comportamento all'apparenza vantaggioso, la particolare dipendenza del PLS dai dati rende difficile esaminarne a fondo le proprietà, dunque dato un particolare problema, non è detto che un'approssimazione di rango basso dia un risultato soddisfacente; al contrario, un gran numero di iterazioni potrebbero essere necessarie per ottenere una riduzione significativa del residuo.

Il PLS è quindi un metodo da utilizzare con estrema attenzione, ma per problemi di taglia moderata, il costo computazionale extra che comporta è marginale rispetto alla bontà del risultato.

6.1 Spunti per un Seminario

Nonostante l'autore dell'articolo si proponga di confrontare il PCR e il PLS, non mostra mai un confronto diretto tra i due, ma si concentra sull'analisi del secondo in maniera approfondita, tralasciando il primo.

Inoltre non esamina nè il costo computazionale, nè le stabilità degli algoritmi che descrive, lasciando per esempio aperta la questione su quale dei due metodi PLS e LBD sia più conveniente usare.

Infine, sostiene che l'LBD sia matematicamente equivalente al metodo del gradiente coniugato, applicato all'equazione normale $A^tAx = A^ty$, rimandando ad altre fonti.

Possibili spunti di sperimentazioni, sarebbero dunque il test degli algoritmi (PCR, PLS, LBD, e Gradiente Coniugato) in termini di tempo, e stabilità numerica, confrontando in particolare la perdità di ortogonalità nelle basi generate, e le norme del residuo generato e della soluzione di rango k.