SVD Factorization and Least Squares

Vediamo l'applicazione di SVD al prob dei minimi quadrati. Come sempre ho il sistema sovradeterminato e lo risolvo nel senso dei minimi quadrati, cioè voglio calcolare x tale che la norma 2 del residuo è minima.
Partiamo dal caso in cui il rango è massimo cioè n. L'SVD è scritta nelle due formule sia come forma classica che nella forma "della diagonalizzazione di A".
L'idea della formula nella slide è che vuole minimizzare quel residuo, la norma due del vettore ottenuto con Ax-b non cambia se moltiplico entrambi con una matrice ortogonale. Uso Ut come matr ortogonale e accanto ad A aggiunge la matricei dentità (V*Vtrasposto), è una sorta di trucco.

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

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

solving $Ax=b$

overdetermined system

$$\min_x \|Ax-b\|_2$$

$$A \in \Re^{m \times n},\ m > n$$

$$x \in \Re^n, b \in \Re^m$$

$rank(A)=n$

$$\min_x \|Ax-b\|_2^2 = \min_x \|U^T Ax - U^T b\|_2^2 = \min_x \|U^T AVV^T x - U^T b\|_2^2$$

Ora quello che faccio è scomporre il prod Ut*b in due pezzi, cioè il prod mi da il vettore e vedo questo vettore scomposto in due pezzi da n e m-n. Quindi riformulo il termine della formula blu, in questo caso aggiungo questa nuova definizione di Ut * b e osservo che Sigma è uguale a quella roba in rosso.

SVD Factorization and Least Squares

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

$$U^T AV = \Sigma$$

$$A \in \Re^{m\times n},\ m > n$$

$$rank(A)=n$$

$$x \in \Re^n, b \in \Re^m$$

$$\min_x \left\|Ax-b\right\|_2^2 = \min_x \left\|U^T Ax - U^T b\right\|_2^2 = \min_x \left\|U^T AVV^T x - U^T b\right\|_2^2$$

$$U^T b = \begin{pmatrix} c \\ d \end{pmatrix} \quad \begin{matrix} n \\ m-n \end{matrix}$$

$$\min_x \left\|Ax-b\right\|_2^2 = \min_x \left\|U^T AVV^T x - U^T b\right\|_2^2 = \min_x \left\|\Sigma V^T x - \begin{pmatrix} c \\ d \end{pmatrix}\right\|_2^2$$

SVD Factorization and Least Squares

Ora Vt*x lo chiama w e sostituisce, sigma è una matr rettangolare diagonale, quindi ha il blocco quadrato diagonale superiore e un blocco nullo sotto. Ora sigma *w diventa quel termine in rosso, ma di questo deve calcolare la norma due al quadrato, cioè la somma dei quadrati quindi la posso scomporre nella somma dei primi n componenti + gli m-n componenti cioè -d, metto +d perché - e + di qualcosa farne la norma è la stessa cosa.
Inoltre notare che d è fuori dal min perchè a noi interessa trovare quel w, quindi d è costante.
Quindi ora per minimizzare quella roba occorre risolvere il ssistema w = sigma C. Sigma è alla - 1, perché bisogna cambiare il segno quindi gli ele sulla diag sono elevati alla -1. Ricordando il valore di V ottengo le ultime due formule in giallo

$$\min_x \|Ax-b\|_2^2 = \min_x \|U^T AVV^T x - U^T b\|_2^2 = \min_x \left\|\Sigma V^T x - \begin{pmatrix} c \\ d \end{pmatrix}\right\|_2^2$$

$$V^T x = w \qquad \Sigma = \begin{pmatrix} \Sigma_n \\ 0 \end{pmatrix} \qquad = \min_w \left\|\Sigma w - \begin{pmatrix} c \\ d \end{pmatrix}\right\|_2^2 =$$

$$= \min_w \left\|\begin{pmatrix} \Sigma_n w \\ 0 \end{pmatrix} - \begin{pmatrix} c \\ d \end{pmatrix}\right\|_2^2 = \min_w \|\Sigma_n w - c\|_2^2 + \|d\|_2^2$$

$$w = \Sigma_n^{-1} c \qquad V^T x_{LS} = \Sigma_n^{-1} c \qquad x_{LS} = V \Sigma_n^{-1} c$$

SVD Factorization and Least Squares

La prima riga di questa slide posso saltarla semplicemente spiega cosa otteniamo facendo sigma meno uno * c cioè un vettore fatto da ci fatto sigmai dove ci = quella roba li nella slide. Xls è quindi = alla riformulazione portando Vt a sx e riscrivendo sigma-1*c (xls però lo possosscivere anche nel modo scritto nel blocco grande ovvero la combinazione lineare delle colonne di V per quelli elementi cioè ci/sigmai.
Xls lo possiamo esprimere in fuznione di B in modo da evidenziare la pseudo inversa.Per fare questo costruisco sigma +, cioè prendo sigma e i valori singolari faccio l'inverso, e il resto rimane uguale inoltre ne faccio la traposta. Dopodichè ricordiamo che U^t *b è = a c, non consideriamo d perchè quella parte viene annullata dal blocco 0 di sigma(nella sua formula generale). Xls ha una formula generica scritta nella slide, nella successiva vediamo la formula con le sue sostituzioni.

$$V^T x_{LS} = \Sigma_n^{-1} c = \begin{pmatrix} c_1/\sigma_1 \\ \vdots \\ c_n/\sigma_n \end{pmatrix}, c_i = u_i^T b$$

$$x_{LS} = V \begin{pmatrix} c_1/\sigma_1 \\ \vdots \\ c_n/\sigma_n \end{pmatrix}$$

$$x_{LS} = \sum_{i=1}^{n} \left(\frac{u_i^T b}{\sigma_i} \right) v_i$$

pseudoinverse:

$$x_{LS} = A^+ b$$

$$U^T b = \begin{pmatrix} c \\ d \end{pmatrix} \Rightarrow U_n^T b = c$$

$$x_{LS} = V \Sigma_n^{-1} U_n^T b$$

$$A^+ = V \Sigma_n^{-1} U_n^T$$

SVD Factorization and Least Squares

Ora sapendo la formula generico di xls possiamo liberare xls dal termine V^t trasposto semplicemente moltiplicando ambo i membri per V e riscrivendo a dx dell'= tutto in funzione delle nuove definizioni quindi xls = V (perchè abbiamo molitplicato V viper fa scompare Vt) sigma^1 t perchè sigma1*a è stata risattata in questo modo INGENERALE FA SOLO RIFORMULAZIONE PER TOGLIERE V^T.
Inoltre aggiunge che la pseudo inversa è quindi tutto il temrine dopo xls = ... per comodità pone quella roba a A+, quindi la pseudoinversa si ottiene facendo la traposta di U* Vt* sigma trasposto e fare l'inversa delle matr diagonali (vedi formula di A+). Quesot è il riassunto di come esprimo la pseudo inversa. Tutto questo è semplice pke con la svd otteniamo subito U^t e V ma anche sigma è quasi immediata pke dobbiamo fare qualche piccola operazione

$$V^T x_{LS} = \Sigma_n^{-1} c = \begin{pmatrix} c_1/\sigma_1 \\ \vdots \\ c_n/\sigma_n \end{pmatrix}, c_i = u_i^T b$$

$$x_{LS} = V \begin{pmatrix} c_1/\sigma_1 \\ \vdots \\ c_n/\sigma_n \end{pmatrix}$$

$$x_{LS} = \sum_{i=1}^{n} \left(\frac{u_i^T b}{\sigma_i} \right) v_i$$

pseudoinverse:

$$x_{LS} = A^+ b$$

$$\Sigma_n^{-1} c = \Sigma^+ U^T b$$

$$\Sigma^+ = \begin{pmatrix} \Sigma_n^{-1} \\ 0 \end{pmatrix}^T$$

$$x_{LS} = V \Sigma^+ U^T b$$

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

reduced SVD factorization

$$A = U_n \Sigma_n V^T \qquad A \in \Re^{m \times n}, m > n$$

$$x \in \Re^n, b \in \Re^m$$

Potremmo usare la SVD ridotta cioè consiuderare solo gli n elementi. Però ricoridamo che possiamo ottnere la soluzione Xls lavorando con i proiettori ortogonali perchè so che la soluzione xls è tale che Axls è la proiezione ortogonale di b sul range A. Poichè io ho una base ortogonale sul range di A dato da Un allora la frase si formalizza con la prima uguaglianza. (Un Tnt è il proiettore ortogonale applicato a b). Al posto di A metto la fattorizzazione SVD ridotta e a questo punto moltiplico ambo i membri da sx per Unt quindi ottengo la formula con rettangolo rosso nella slide successiva, che è esattamente la stessa cosa di ciò che ottengo nel caso precedente. Nel blocco grigio in basso a dx in questa slide è il passaggio intermedio prima del blocco rosso della slide successvia
Nella slude successiva c'è anche l'altro rettangolo rosso che è l'ressressione della pseduo inversa in funzione di Un e sigma n.

Least Squares full rank

$$\min_x \|Ax - b\|_2$$

$$rank(A) = n$$

the solution x_{LS} is such that Ax_{LS} is the orthogonal projection of b onto the $range\,(A)$

$$Ax_{LS} = U_n U_n^T b$$

$$U_n \Sigma_n V^T x_{LS} = U_n U_n^T b$$

$$\Sigma_n V^T x_{LS} = U_n^T b$$

$$x_{LS} = \left(\Sigma_n V^T\right)^{-1} U_n^T b$$

reduced SVD factorization

$$A = U_n \Sigma_n V^T$$

$$A \in \Re^{m \times n},\ m > n$$

$$x \in \Re^n, b \in \Re^m$$

Least Squares full rank

$$\min_x \|Ax - b\|_2$$

$rank(A)=n$

the solution x_{LS} is such that Ax_{LS} is the orthogonal projection of b onto the $range\ (A)$

$$Ax_{LS} = U_n U_n^T b$$

$$U_n \Sigma_n V^T x_{LS} = U_n U_n^T b$$

$$x_{LS} = V \Sigma_n^{-1} U_n^T b$$

$$A^+ = V \Sigma_n^{-1} U_n^T$$

Quindi l'algoritmo è questo:
calcolare la fatt ridotta di svd
calc il vettore c
risolvere il sist diagonale
ricostruisci x come V*w

reduced SVD factorization

$$A = U_n \Sigma_n V^T$$

$$A \in \Re^{m \times n},\ m > n$$

$$x \in \Re^n, b \in \Re^m$$

Least Squares full rank

$$\min_x \|Ax - b\|_2$$

$rank(A)=n$

Algorithm for Least Squares

1. Compute the reduced SVD factorization of A
2. Compute the vector $c = U_n^T b$
3. Solve the diagonal system $\Sigma_n w = c$
4. Set $x_{LS} = Vw$

$$A = Q_n R_n$$
$$c = Q_n^T b$$
$$R_n x_{LS} = c$$

```
[Un,Sn,Vn] = svd(A,0);
xls = Vn * ((Un'*b)./diag(Sn))
```

Ora affrontiamo il prob del minimi quadrati con il caso no full rank, qui si parla di Rank deficient perchè è minore di N. In questo problema esistono infinite Xls ma solo una ha lunghezza minima. Quindi procediamo in questo modo, A è = alla fattorizzazione economia con r, allora seguo la strada della proiezione ortogonale (Axls è la proiezione ortogonale) cioè faccio quella formuula UrUrtb, quindi invece di A metto la fattorizzazione (quel blocco in blu che è un sistema sottodeterminato), chiaramente si tratta di un sistema che ha infinite soluzioni, quello che si fa qui è moltiplicare ambo i membri prima per sigma^-1 e poi * Vr così tolgo gli elementi vicino a xls. Otteniamo così la formula finale cioè il blocco rosso a sx. I lblocco a sx rosso invece è la risoluzioni che è quello che dobbiamo trovare, cioè quello di lunghezza minima. A+ è definita in questo modo quindi xls = A+b

Least Squares

rank deficient

$$\min_x \|Ax-b\|_2$$

$rank(A) = r < n$

- ✓ there are infinite solutions to the minimum problem
- ✓ there is only one solution x_{LS} of minimum length

$$A = U_r \Sigma_r V_r^T \qquad Ax_{LS} = U_r U_r^T b$$

$$\Sigma_r V_r^T x_{LS} = U_r^T b \qquad V_r V_r^T x_{LS} = V_r \Sigma_r^{-1} U_r^T b$$

underdetermined!

since we are seeking the solution of miminum length, it holds that

$V_r V_r^T x_{LS} = x_{LS}$, so that we have

$$x_{LS} = V_r \Sigma_r^{-1} U_r^T b$$

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

Credo che l'ha messa quest'anno ma non ci sono video dove spiega il significato.

Least Squares pseudoinverses

$$\min_x \|Ax - b\|_2 \qquad x_{LS} = A^+ b$$

$$A^+ = \left(A^T A\right)^{-1} A^T b$$

Normal equations

$$A^+ = R_n^{-1} Q_n^T b$$

QR factorization

$$A^+ = V \Sigma_n^{-1} U_n^T$$

SVD factorization

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

SVD factorization

Vediamo ora l'uso della SVD nella "latent semantic analisis". Abbiamo la matrice B di termini - documenti, e ne facciamo l'SVD, in particolare approssimiamo B con una matrice B^k di rango K (SVD k-troncata), infine uso le colonne di B^k nel calcolo della similarità

SVD Factorization and LSA

SVD of the terms – documents matrix B

- ✓ SVD of B (terms – documents matrix)
- ✓ approximate B with a matrix $B^{(k)}$ of rank k (for a suitable k)
- ✓ use the columns of $B^{(k)}$ in computing the similarity

Lo spazio delle colonne ha a che fare con i documenti mentre lo spazio delle righe con i termini, il resto già l'ho detto prima. A sx è la matrice B orginale e a dx è con la k truncated SVD OF B

SVD Factorization and LSA

information on both the
column space (**documents**) and the
row space (**terms**)

rank reduction

consider the matrix $B^{(k)}$, for a suitable k, which is the best rank k approximation of B

$$B = U\Sigma V^T$$

$$B^{(k)} = U_k \Sigma_k V_k^T$$

k-truncated SVD of B

Arriva una query q che è un vettore che sta nel range di B, calcolo la similarità come il coseno dell'angolo tra q e ogni colonna di B. (per trovare le colonne usa ej che è un vettore con tutti 0 e un 1 sulla j esima componente per cui noi vogliamo la j esima colonna, infatti fai il prod scalare con B e ottieni tutti 0 tranne i valori della j esima colonna di b. Ovviamente al posto di B mette la sua fattorizzazione, al passaggio successivo fa il trasposto del prodotto di sopra. U^t * q vuol dire proiettare la query sulla nuova base Uk che ha una dimensione molto minore del numero delle parole chiave. Poi fare sigma k * vk e moltiplicare * ej vuol dire che ne selezioniamo la j-esima riga che chiamiamo sj. Sj quindi è il vettore documento scalato. Si chiamano così e di fatto sono le colonne di v trapostok scalae di sigma k

SVD Factorization and LSA

query q

compute **similarity**

$$\cos(\theta_j) = \frac{\left(B^{(k)} e_j\right)^T q}{\left\|B^{(k)} e_j\right\|_2 \|q\|_2} = \frac{\left(U_k \Sigma_k V_k^T e_j\right)^T q}{\left\|U_k \Sigma_k V_k^T e_j\right\|_2 \|q\|_2} =$$

j-th column of $B^{(k)}$

$$= \frac{e_j^T V_k \Sigma_k \left(U_k^T q\right)}{\left\|\Sigma_k V_k^T e_j\right\|_2 \|q\|_2} =$$

scaled document vector

$$s_j = \Sigma_k V_k^T e_j \equiv \left(\Sigma_k V_k^T\right)_j$$

Quindi alla fine ci troviamo a calcolare questo coseno. A questo punto è opprtuno notare che non è necessario calcolare esplicitamente B^k perchè a me serve solo Uk che è una matrice mxk quindi non mi serve la amtrice mxn. Inoltre la norma 2 al denominatore può essere precalcolata quindi non serve farlo per ogni s_j

SVD Factorization and LSA

query q

scaled document vectors

$$\Sigma_k V_k^T \equiv \left(s_1, s_2, \ldots, s_n\right)$$

compute **similarity**

$$\cos\left(\theta_j\right) = \frac{s_j^T \left(U_k^T q\right)}{\left\|s_j\right\|_2 \left\|q\right\|_2}$$

- ✓ it is not necessary to explicitly compute $B^{(k)}$
- ✓ the 2-norm of scaled documents can be precomputed, stored and used for subsequent queries

Le k componenti del vettore sj sono le coordinate della j-sima colonna di B^k sulla base definita dalle colonne di Uk. Ignora i calcoli.

SVD Factorization and LSA

query q

scaled document vectors

$$\Sigma_k V_k^T \equiv (s_1, s_2, \ldots, s_n)$$

compute **similarity**

✓ the k components of the vector s_j are the coordinates of the j-th column of $B^{(k)}$ on the basis formed by the columns of U_k

$$B^{(k)} = U_k \Sigma_k V_k^T = U_k (s_1, s_2, \ldots, s_n)$$

$$\left(B_1^{(k)}, B_2^{(k)}, \ldots, B_n^{(k)}\right) = (U_k s_1, U_k s_2, \ldots, U_k s_n)$$

Questa slide riporta quello detto prima, quindi di fatti i coseni sono calcolati sulla nuova base equindi questi coseni coinvolgono la rappresenzatione del j-simo documento sulla base Uk e la rappresentazione della query sulla base Uk ma va avanti (la riferimento al numeratore)

SVD Factorization and LSA

query q

scaled document vectors

$$\Sigma_k V_k^T \equiv \left(s_1, s_2, \ldots, s_n\right)$$

✓ the k components of the vector $U_k{}^T q$ are the coordinates, respect to the basis formed by the columns of U_k, of the projection $U_k U_k{}^T q$ of the query q onto the column space of $B^{(k)}$

$$\cos\left(\theta_j\right) = \frac{s_j^T \left(U_k^T q\right)}{\left\|s_j\right\|_2 \left\|q\right\|_2}$$

is computed respect to the orthogonal basis, formed by the columns of U_k, of the column space of $B^{(k)}$

Le colonne di Vk traposto sono i vettori che rappresentano i documenti nella nuova base, mentre i termini della nuova base sono le righe di Uk (chiaramente in etrambi i casai i valori si devono scalare per fattori opportuni), ma le righe di Uk sono la rappresentazione dei termini sulla base Uk. Quindi se vogliamo rappresentare una colonna di Bk dobbiamo usare come base Uk, mentre se vogliamo rappresentare una riga di Bk dobbiamo usare la colonne di Vk con il vettore dei termini presenti nella corrispondente riga di Uk

in LSA, SVD (k-truncated) factorization allows each document and each term to be represented by a vector in a k-dimensional space, using left and right singular vectors as components

columns of V_k^T (rows of V_k) → document vectors (unscaled!) (coefficients of documents on the orthogonal basis of documents)

rows of U_k → term vectors (unscaled!) (coefficients of the terms on the orthogonal basis of the terms)

$$B^{(k)} = \left(U_k \Sigma_k\right) V_k^T$$

Qui c'è la trattazione che è la stessa slide che ha mostrato quando ha parlato della SVD completa, qui è riportata con la questione K. Alla fine otteniamo il vettore termini scalati, non aggiunge altro.

$$U_k \Sigma_k = \left(\sigma_1 u_1, \ldots, \sigma_k u_k\right) = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_m^T \end{pmatrix}$$

$$B^{(k)} \equiv \begin{pmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_m^T \end{pmatrix} = \begin{pmatrix} y_1^T V_k^T \\ y_2^T V_k^T \\ \vdots \\ y_m^T V_k^T \end{pmatrix}$$

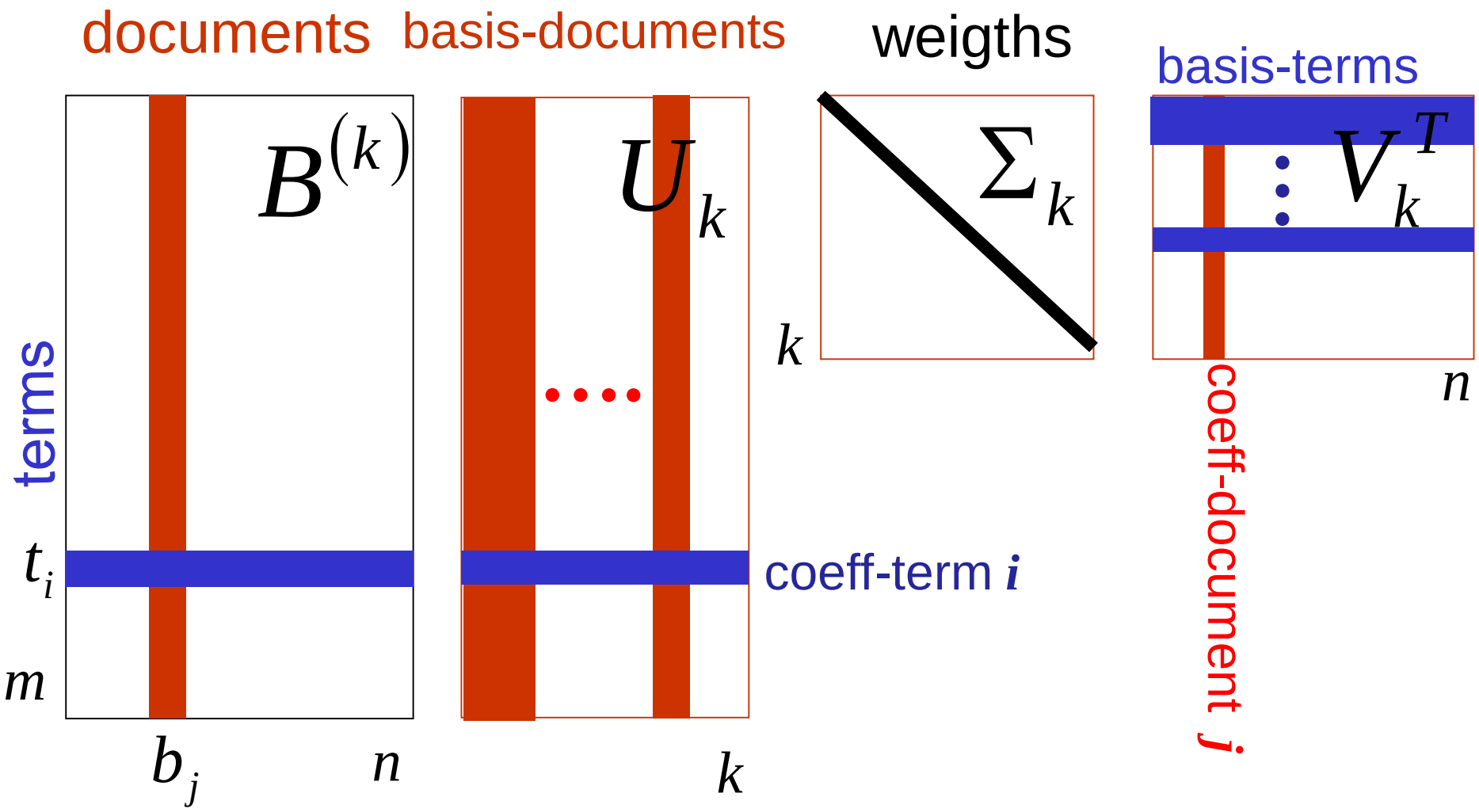
$$t_i^T = y_i^T V_k^T$$

scaled terms vectors

$$U_k \Sigma_k = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_m^T \end{pmatrix}$$

Questa è l'immagine da tenere a mente, vogliamo il k-esimo dovumento di B? (bk) e lo si esprime sulla base delle colonne Uk, quali sono le componenti di questo vettore bj espresso rispetto alla base Uk? Devo prendere la j-esima col di Vkt o j-esima riga di Vk. in sigma ci sono i fattori di scalatura. Se sono interessato ai termini quinidi alle righe blu (tj di B), dove la voglio esprimere questa ? Rispetto alla base dei termini che sono le righe di Vk traposto, quali sono l componenti del vettore ti rispetto alla base Uk, allora devo prendere la riga i-sima Uk il tutto sempre scalato rispetto a sigma k .

$$B^{(k)} = U_k \Sigma_k V_k^T$$



Ora fa un esempio della latent semantic analysis

Label	Titles
B1	A Course on Integral Equations
B2	Attractors for Semigroups and Evolution Equations
B3	Automatic Differentiation of Algorithms: Theory, Implementation, and Application
B4	Geometrical Aspects of Partial Differential Equations
B5	Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra
B6	Introduction to Hamiltonian Dynamical Systems and the N-Body Problem
B7	Knapsack Problems: Algorithms and Computer Implementations
B8	Methods of Solving Singular Systems of Ordinary Differential Equations
B9	Nonlinear Systems
B10	Ordinary Differential Equations
B11	Oscillation Theory for Neutral Differential Equations with Delay
B12	Oscillation Theory of Delay Differential Equations
B13	Pseudodifferential Operators and Nonlinear Partial Differential Equations
B14	Sinc Methods for Quadrature and Differential Equations
B15	Stability of Stochastic Differential Equations with Respect to Semi-Martingales
B16	The Boundary Integral Approach to Static and Dynamic Contact Problems
B17	The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory

Questa è la matrice dei termini documenti

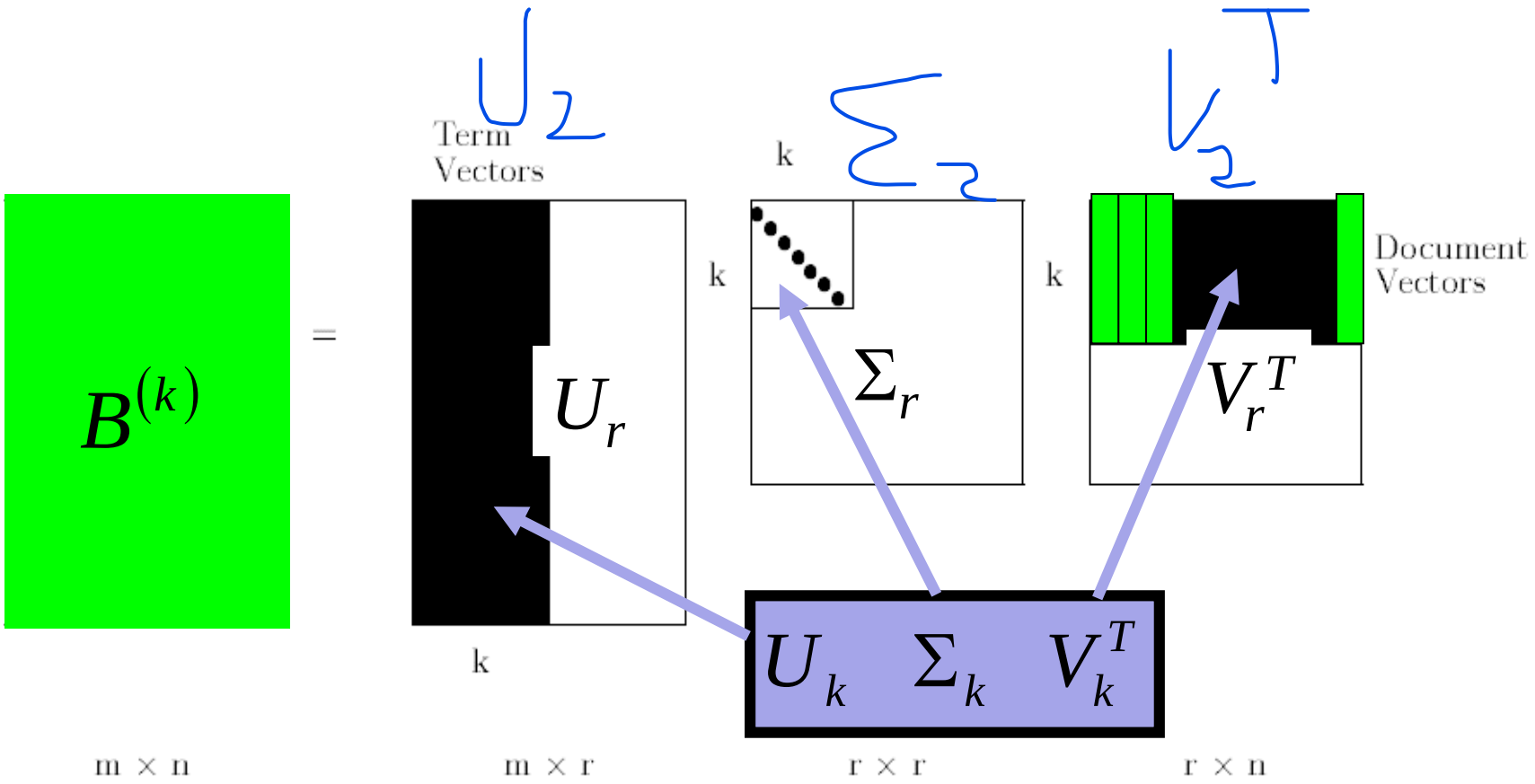
Terms	Documents																
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
algorithms	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
application	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
delay	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
differential	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
equations	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
implementation	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
introduction	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
nonlinear	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
ordinary	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
oscillation	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
partial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
problem	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
systems	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
theory	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

use a truncated SVD with $k = 2$

Applichiamo l'svd troncata con k = 2

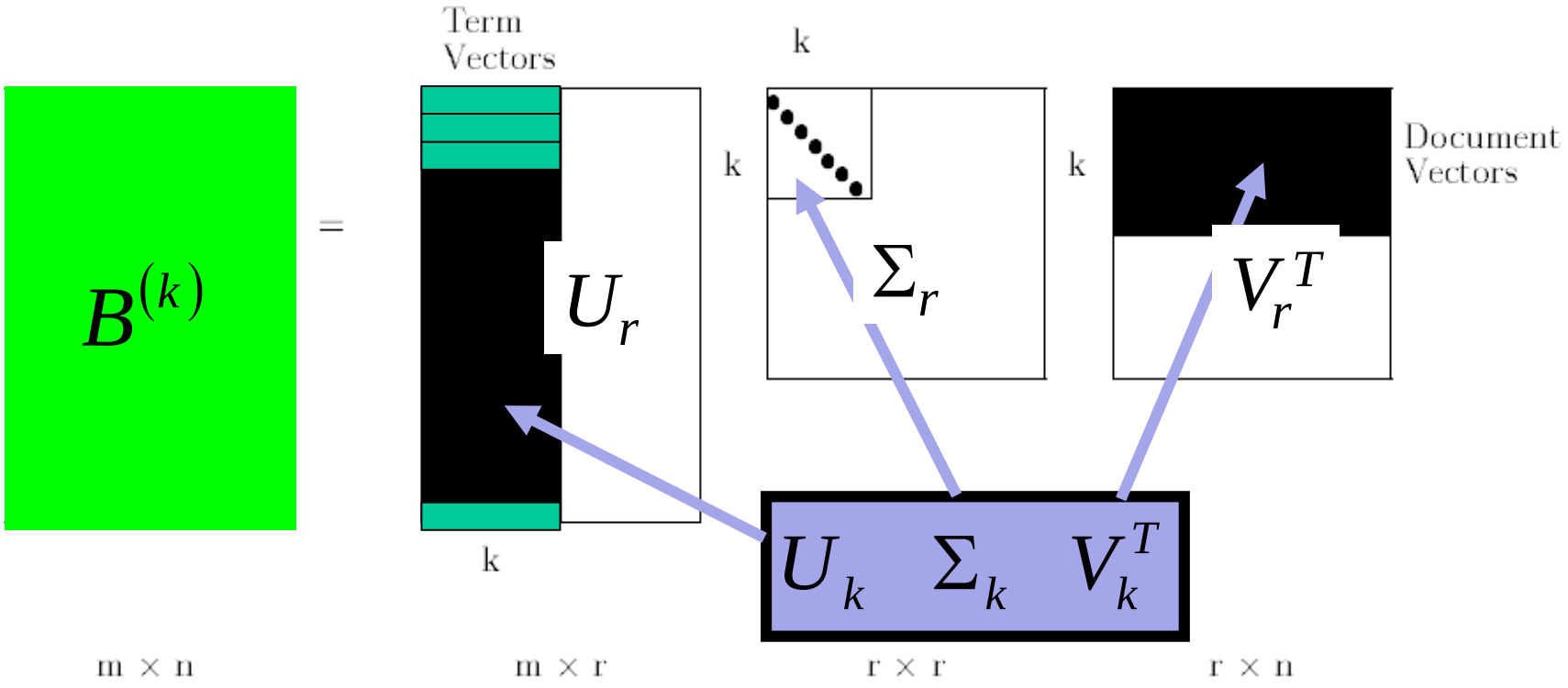
Qui è come sono organizzate le matrici. O mostra che in verde sono i vetotri documenti.

k-truncated SVD with $k = 2$

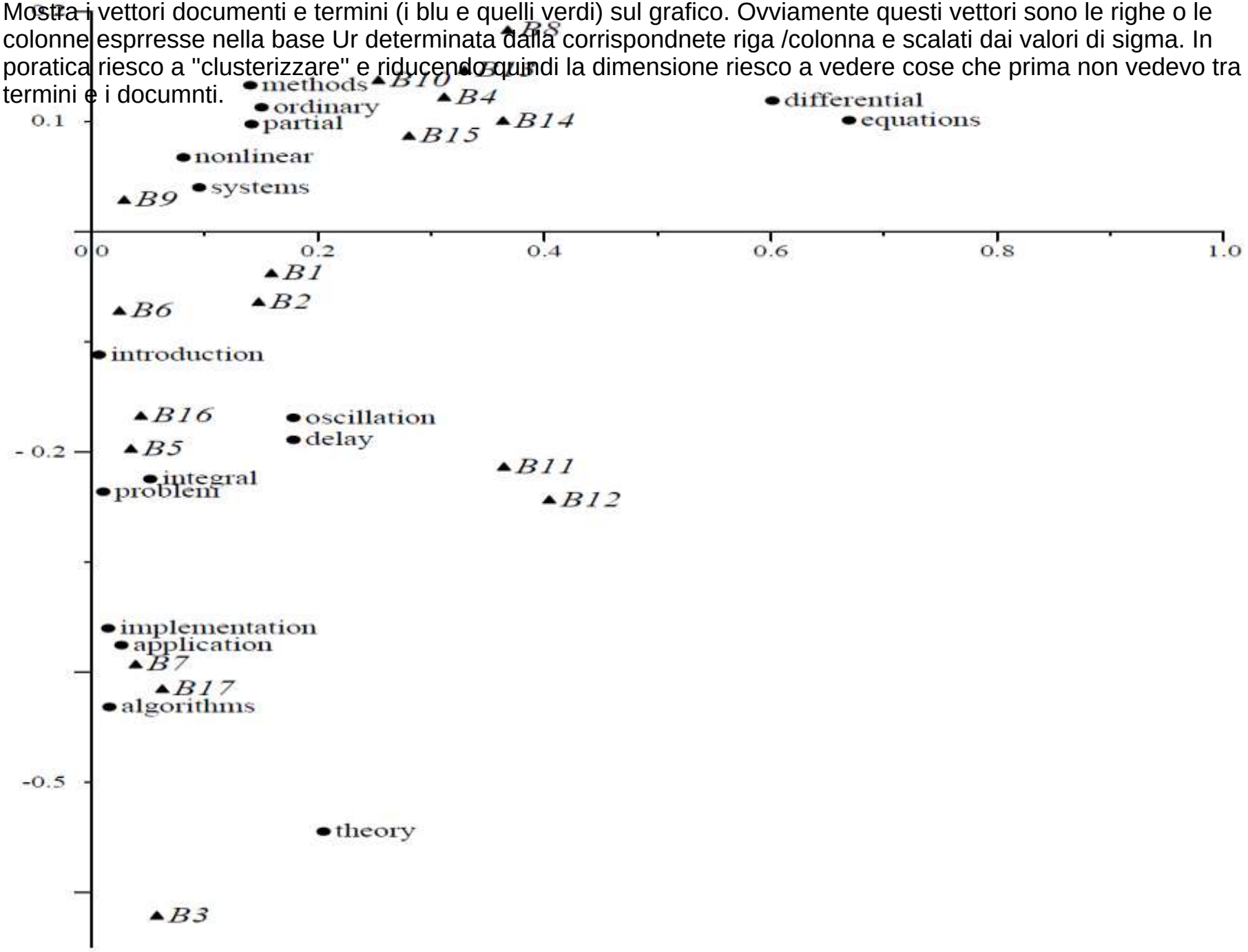


i termini solo le righe azzure

k-truncated SVD with $k = 2$



Mostra i vettori documenti e termini (i blu e quelli verdi) sul grafico. Ovviamente questi vettori sono le righe o le colonne esprresse nella base Ur determinata dalla corrispondnete riga /colonna e scalati dai valori di sigma. In poratica riesco a "clusterizzare" e riducendo quindi la dimensione riesco a vedere cose che prima non vedevo tra i termini e i documnti.

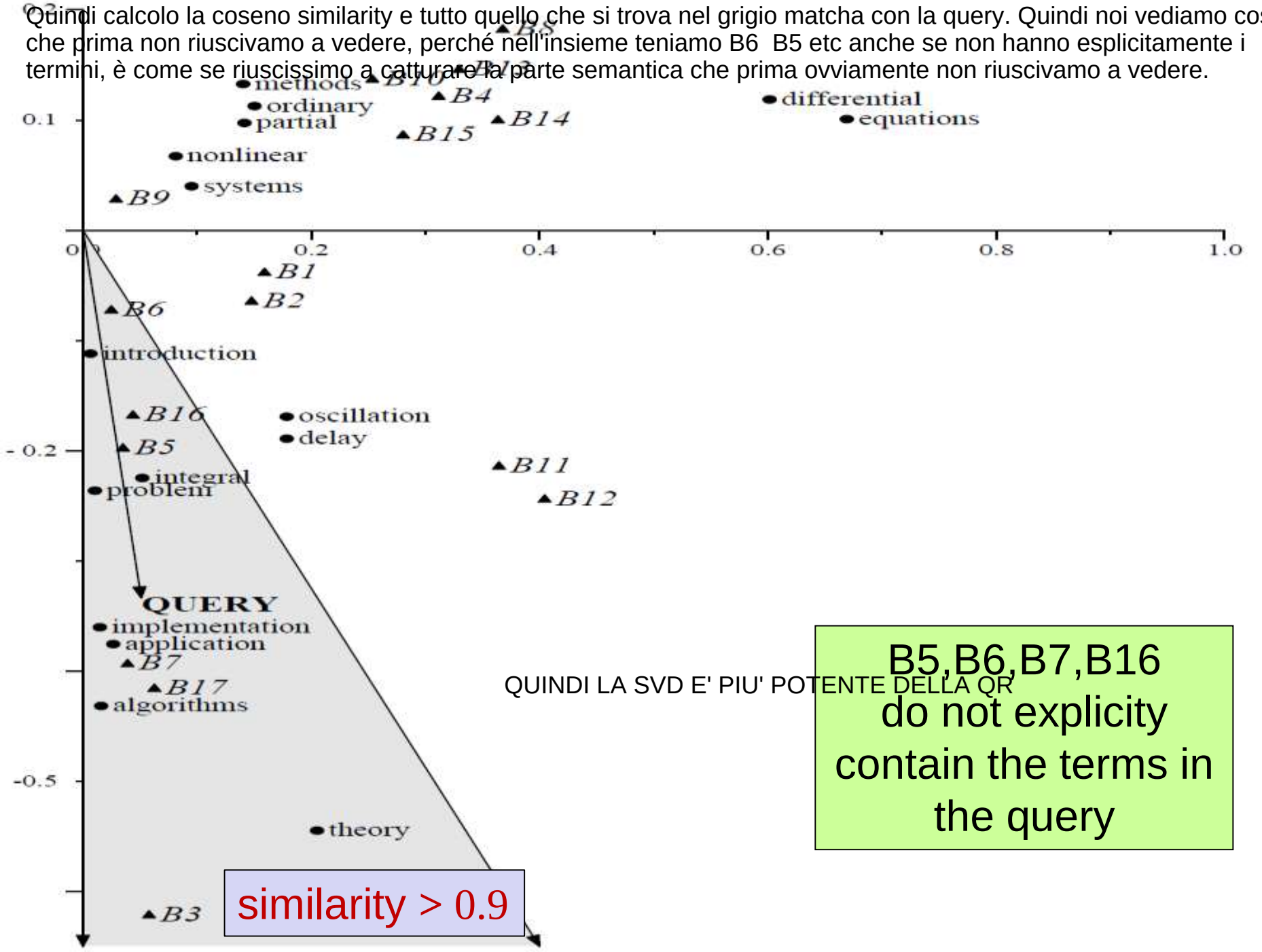


Ora vediamo cosa accade con la query che è un vettore nello spazio di prima, allora calcolo la similarità dell'angolo tra i vettori rispeotto alla query. Supponiamo che questa è la query: application theroy, uella con tutti i numeri è la rappresentazione originale, mentre quella con tutti 0 e 1 è la rappresentazione 2

query: application theory

$$\hat{q} = U_k^T q = \begin{pmatrix} .016 & .027 & .178 & .601 & .670 & .015 & .052 & .007 & .015 & .081 & .150 & .178 & .141 & .010 & .095 & .205 \\ -.431 & -.376 & -.170 & .119 & .120 & -.360 & -.225 & -.112 & .113 & .067 & .113 & -.169 & .097 & -.237 & .040 & -.545 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Quindi calcolo la coseno similarity e tutto quello che si trova nel grigio matcha con la query. Quindi noi vediamo cose che prima non riuscivamo a vedere, perché nell'insieme teniamo B6 B5 etc anche se non hanno esplicitamente i termini, è come se riuscissimo a catturare la parte semantica che prima ovviamente non riuscivamo a vedere.



QUINDI LA SVD E' PIU' POTENTE DELLA QR

B5,B6,B7,B16 do not explicity contain the terms in the query

Ora passiamo alla compressione delle immagini, intesa come riduzione della dimensionaità e anche qui si applica la svd k troncata. Considero la foto rappresentata dalla matrice A per la quale calcolo la SVD, dopodichè calcolo la k troncata che sappiamo essere la somma delle prime k e Ak sappiamo essere la migliore approssimazione di rango k di A Ei cioè matrici di rango

SVD Factorization and image compression (dimension reduction)

matrix A

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

$$A^{(k)} = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} E_i$$

$$E_i = \sigma_i u_i v_i^T$$

SVD è usato molto nella compressione delle immagini ed è vista come un caso particolare della decompositizione di quei tizi nella slide. Il resto basta leggerlo.

SVD Factorization and image compression (dimension reduction)

Lossy compression (Karhunen-Loeve decomposition)

A color RGB image of mxn pixels is represented by means of 3 mxn matrices (red, green and blue intensity, with entries from 0 to 255)

Using SVD, the “best" approximation of rank k is determined for each of the three matrices.

This representation requires, for each matrix, the storage of a number of elements given by $k(m + n + 1)$, instead of mxn.

As k increase, the image quality improves, but the memory space increases

Questo è un esempio matlab che farà

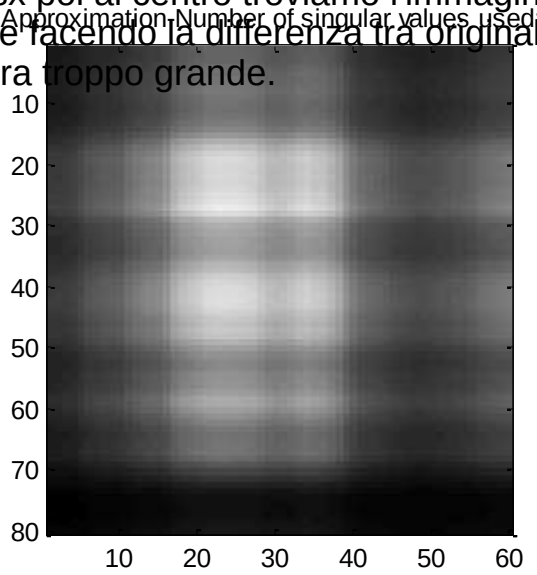
SVD Factorization and image compression (dimension reduction)

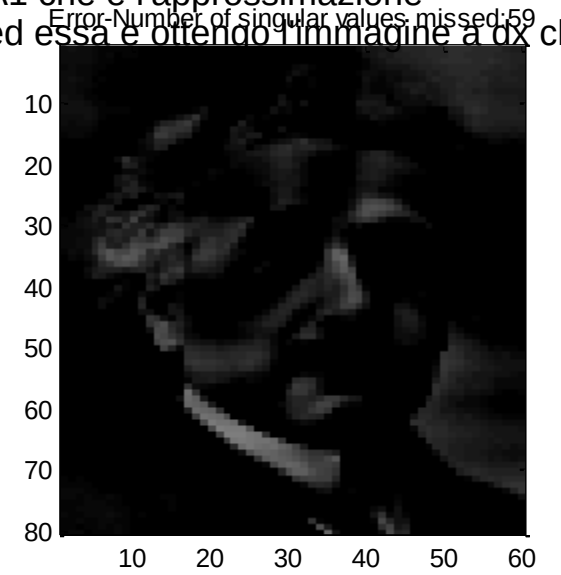
(script: **SVDimage.m**)

$$E_i = \sigma_i u_i v_i^T$$

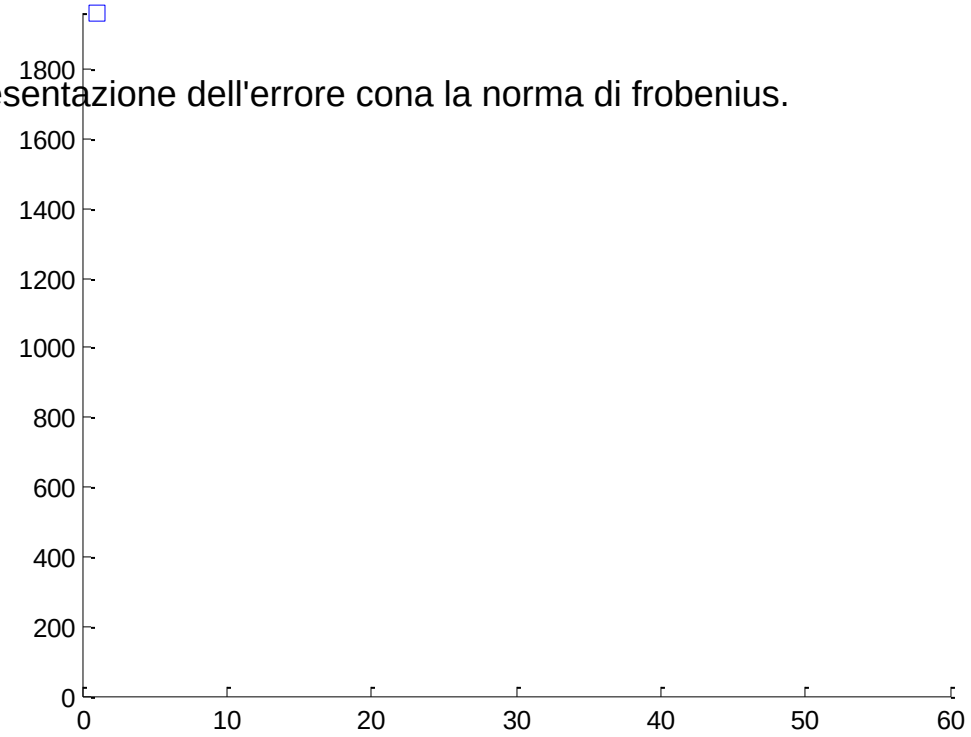
```
 A = imread('einstein.gif');
 A = double(A); A =63*A/max(max(A));A=64-A;
[U,S,V] = svd(A);
 N = min(size(S));
 sing_val = diag(S);
 for k=1:N
  A_appr=A_appr+sing_val(k)*U(:,k)*V(:,k)';
  image(A_appr)
  A_err = A-A_appr;
 end
```

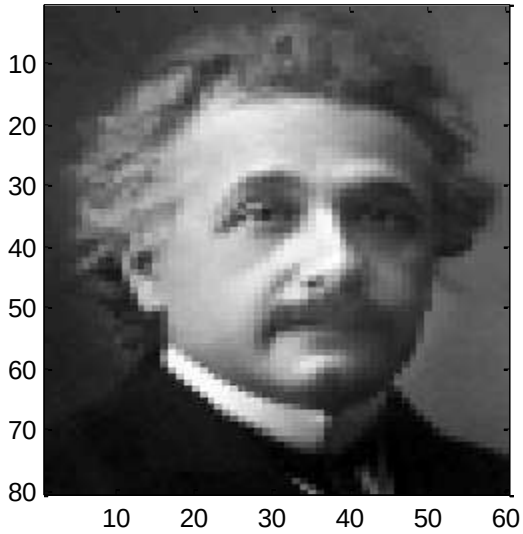
Il riusltato è questo, l'immagine in input a sx poi al centro troviamo l'immagine A1 che è l'approssimazione dell'immagine originale e ne calcolo l'errore facendo la differenza tra originale ed essa e ottengo l'immagine a dx che è abbastanza chiara quindi l'errore è ancora troppo grande.



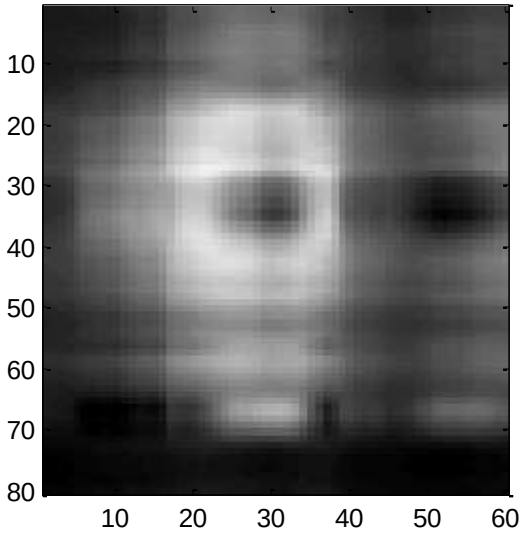


Il grafico sotto è la rappresentazione dell'errore cona la norma di frobenius.

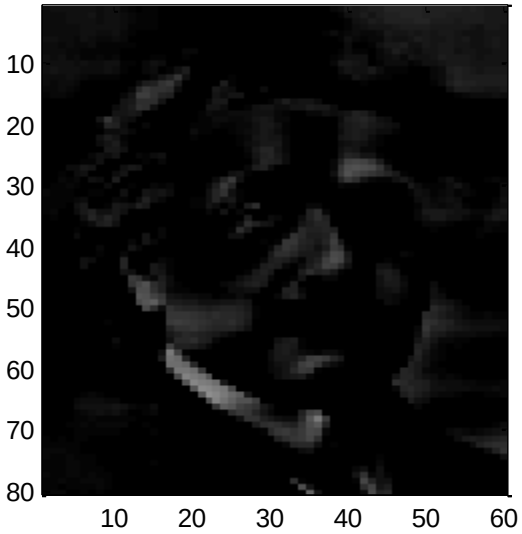




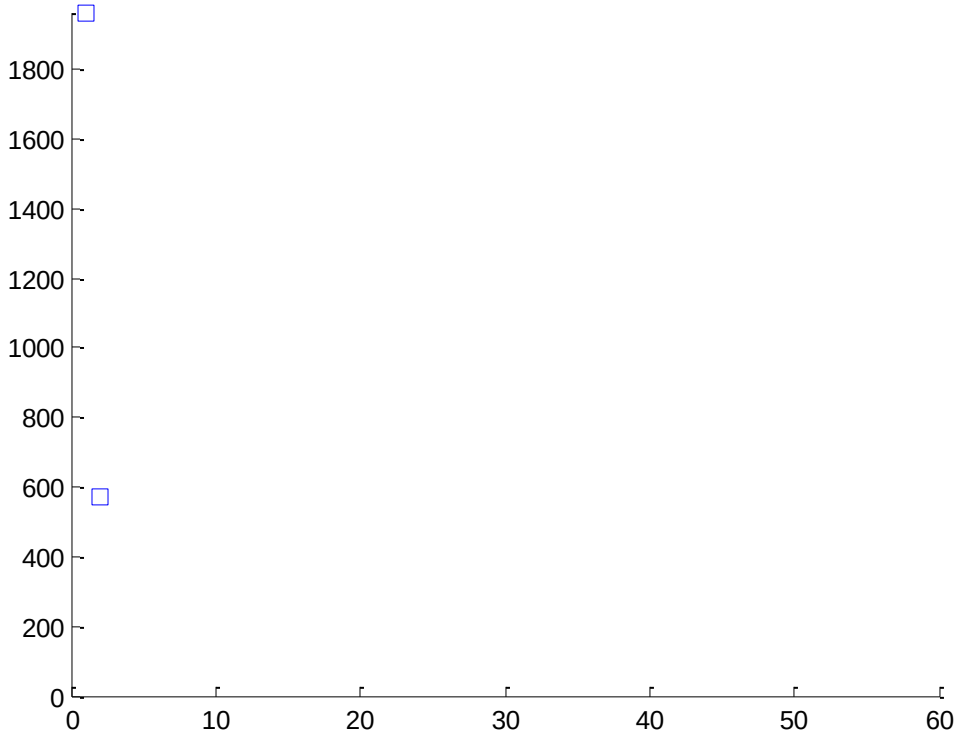
Approximation-Number of singular values used:2

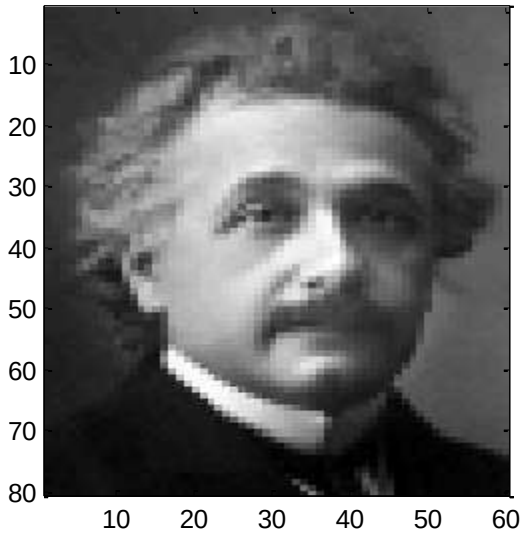


Error-Number of singular values missed:58

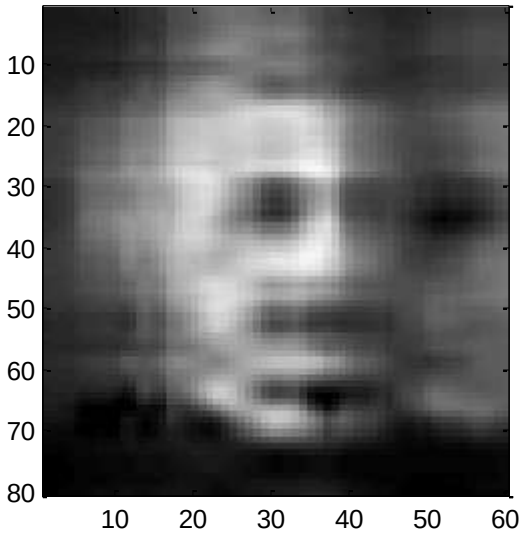


Frobenius norm of approximation error

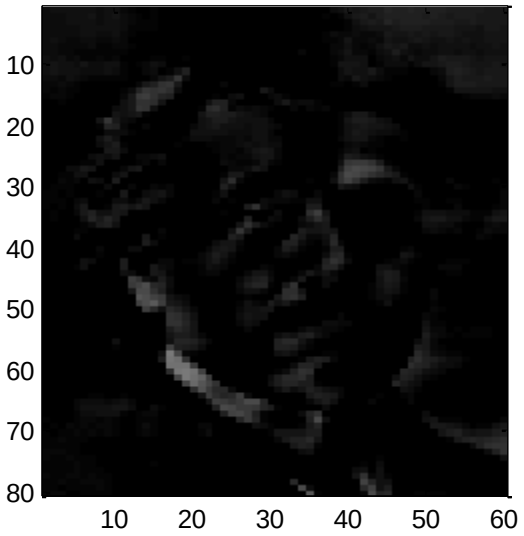




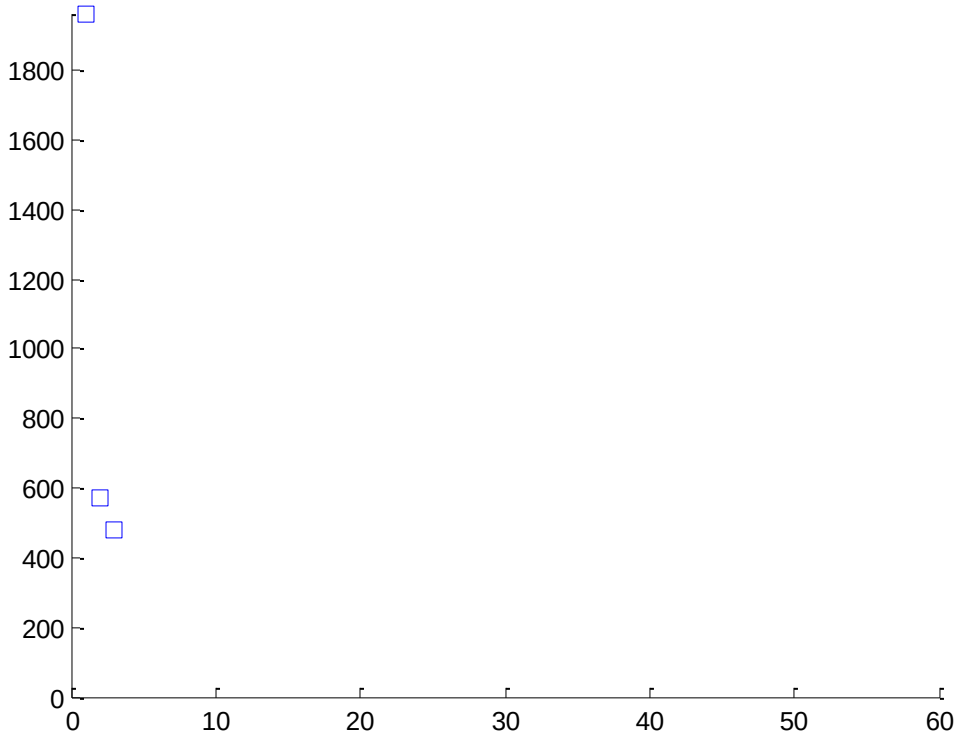
Approximation-Number of singular values used:3

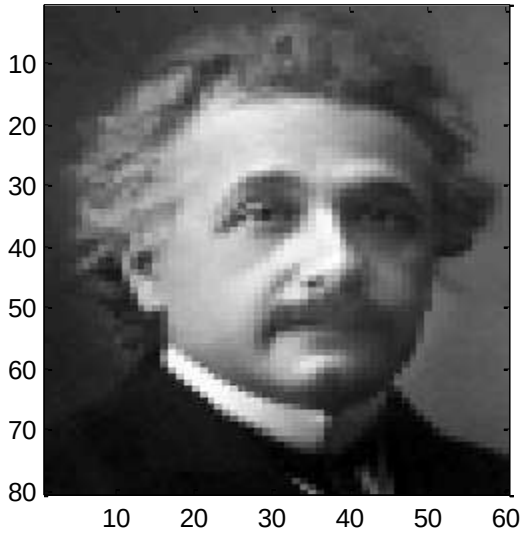


Error-Number of singular values missed:57

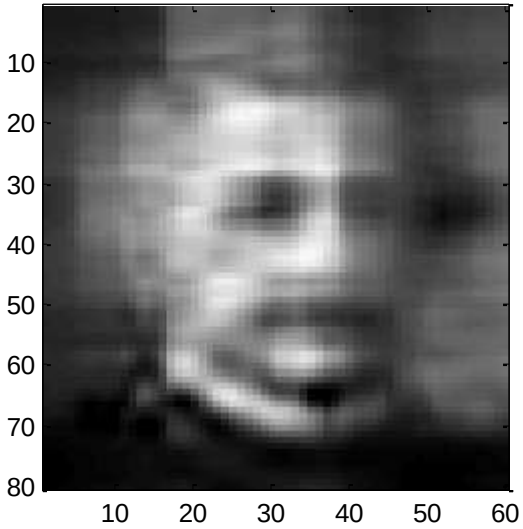


Frobenius norm of approximation error

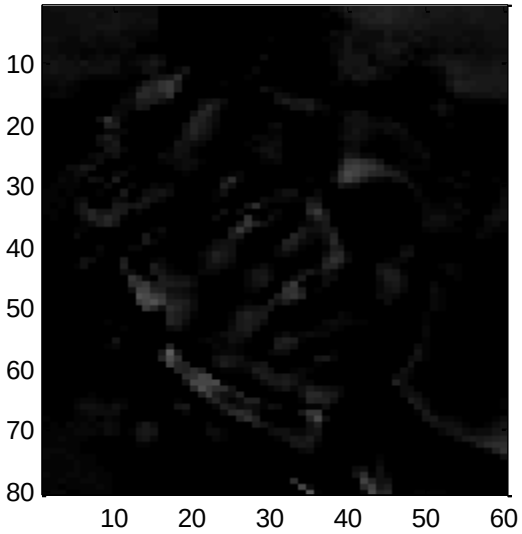




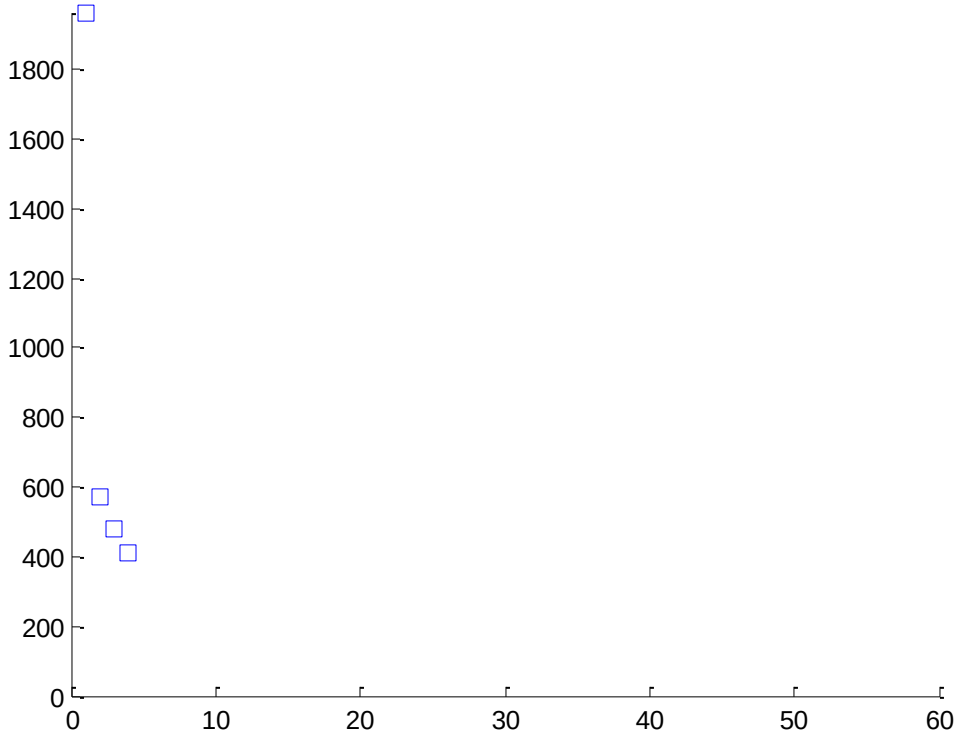
Approximation-Number of singular values used:4

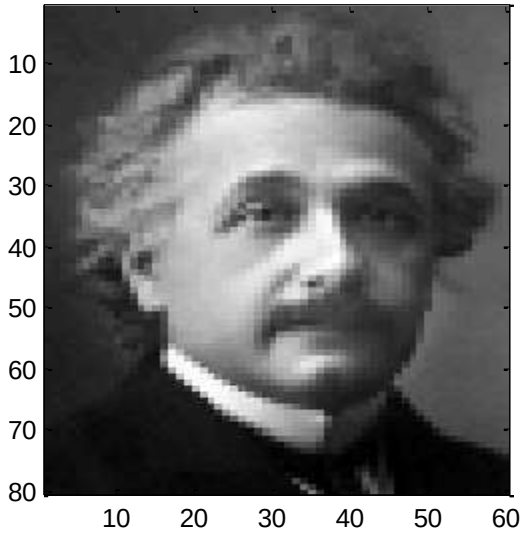


Error-Number of singular values missed:56

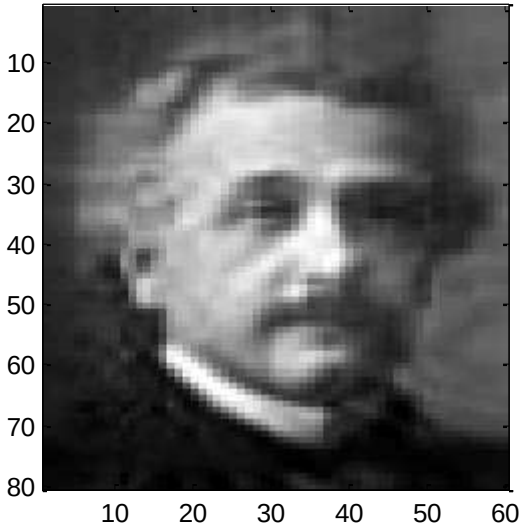


Frobenius norm of approximation error

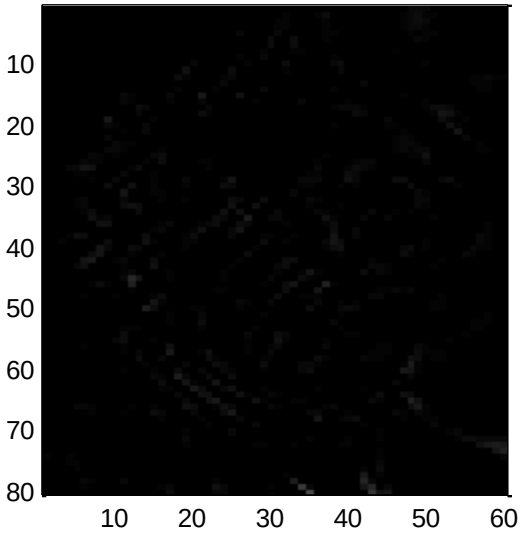




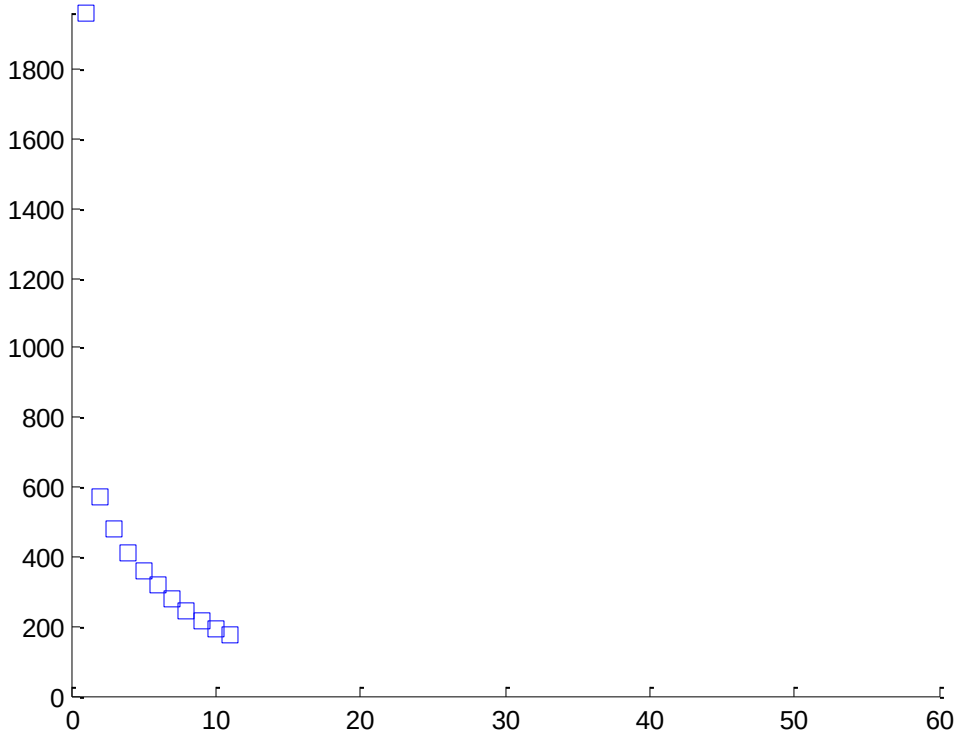
Approximation-Number of singular values used:11



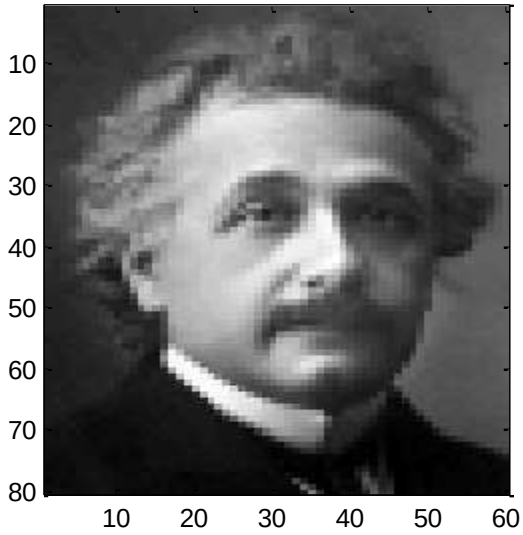
Error-Number of singular values missed:49

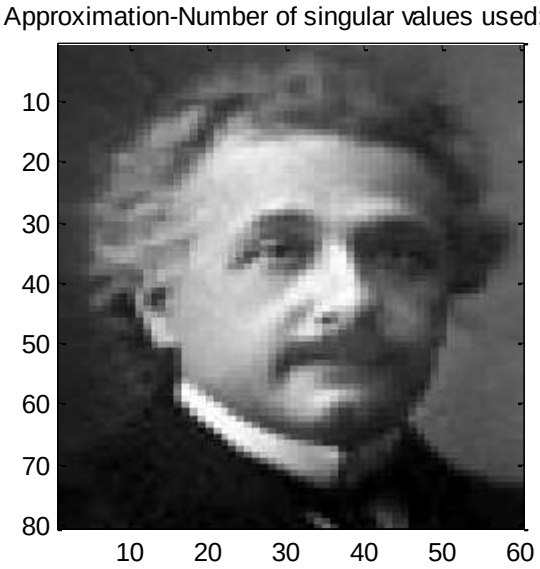


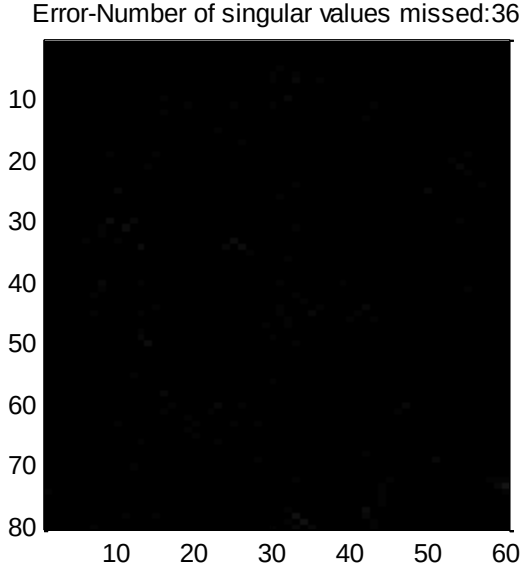
Frobenius norm of approximation error

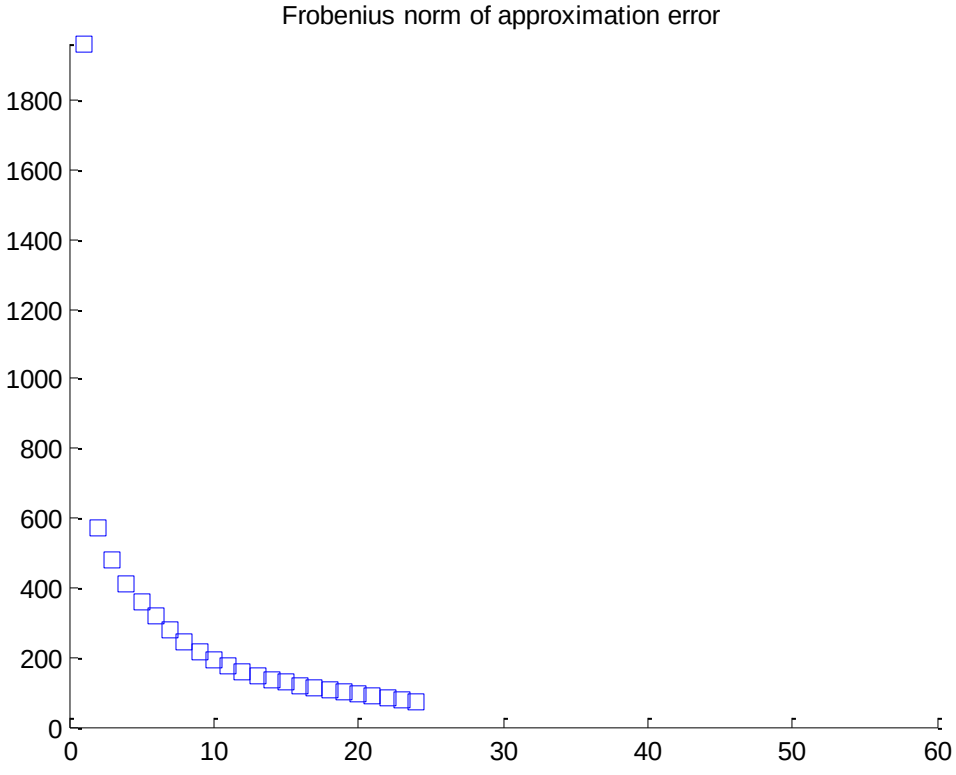


Questa è una buona approssimazione.









Questo è l'algoritmo come prima ma applicato ad un testo scritto.

SVD Factorization and image compression (dimension reduction)

```
text_in = text(0.05,0.5,...
'INFORMATICA APPLICATA (ML e BD)','fontsize',30);
axis off
saveas(gcf,'infoapp.png');
A = imread('infoapp.png');
A = double(rgb2gray(A));
figure(1)
imagesc(A), colormap gray
disp('image size')
[m,n] = size(A);
sprintf('nrows = %d  ncolumns = %d',m,n)
```

```
image size

    nrows = 656  ncolumns = 1493
```


INFORMATICA APPLICATA (ML e BD)
100
200
300
400
500
600
200
400
600
800
1000
1200
1400

L'imm orginale era di rango 62

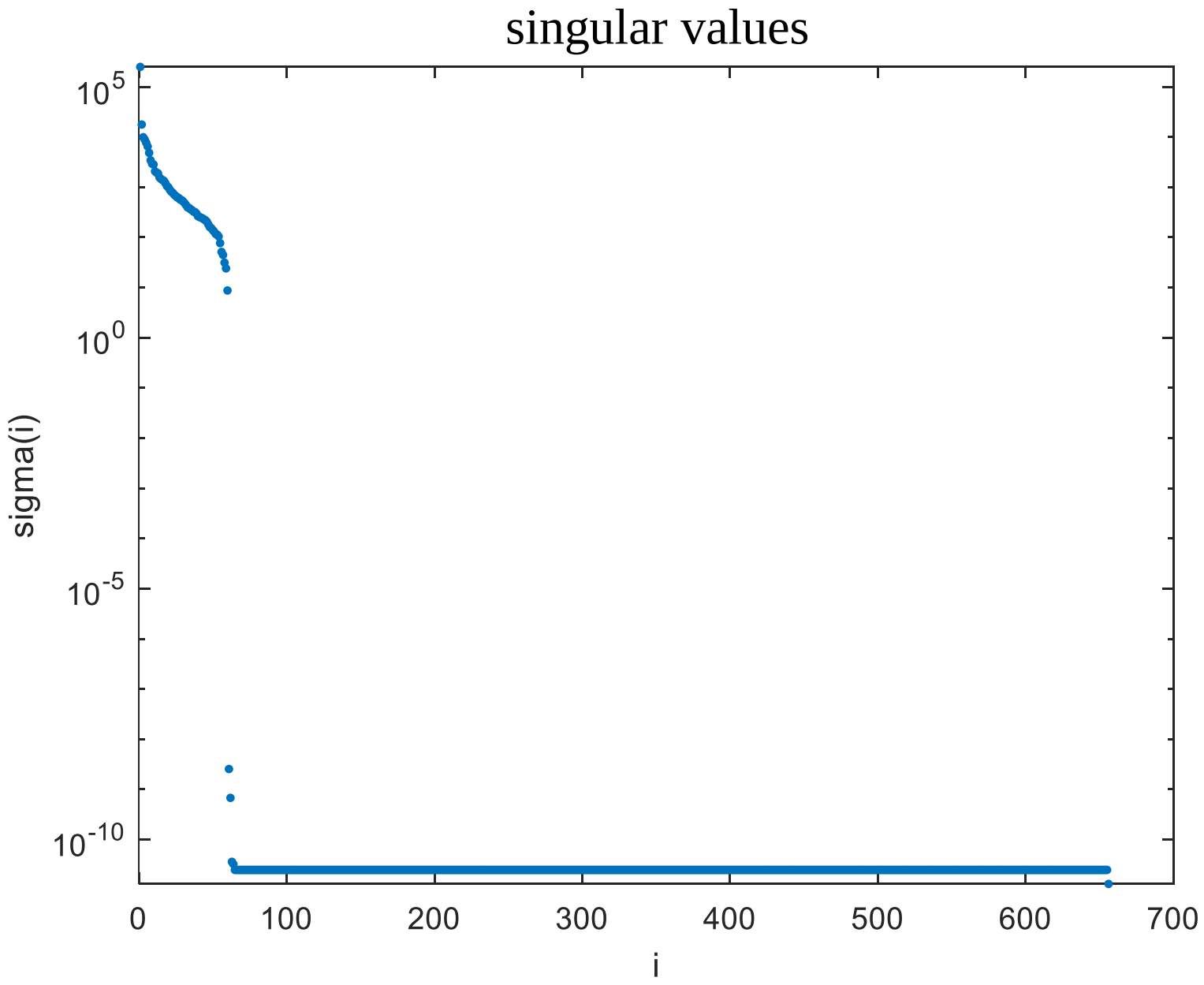
```
[U,S,V] = svd(A);
sigma = diag(S);
figure(2)
semilogy(sigma,'.', 'markersize',8)
title('singular values')
xlabel('i'), ylabel('sigma(i)')
rank_m = find(sigma/sigma(1) > 10*eps,
1,'last')
```

```
rank_m =

    62
```

```
>> length(sigma)
ans =
   656
```

L'erorre tende a 0 moltom molto rapidament.e



```
figure(3)
for i=1:4
    subplot(2,2,i)
    k = 2*i;
    Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
    imagesc(Ak), colormap gray
    axis off
    title(sprintf('rank = %d',k))
end
```

Già invece usando rango 4 riesco a capire ciò che è scritto.

rank=2

rank=4

rank=6

rank=8

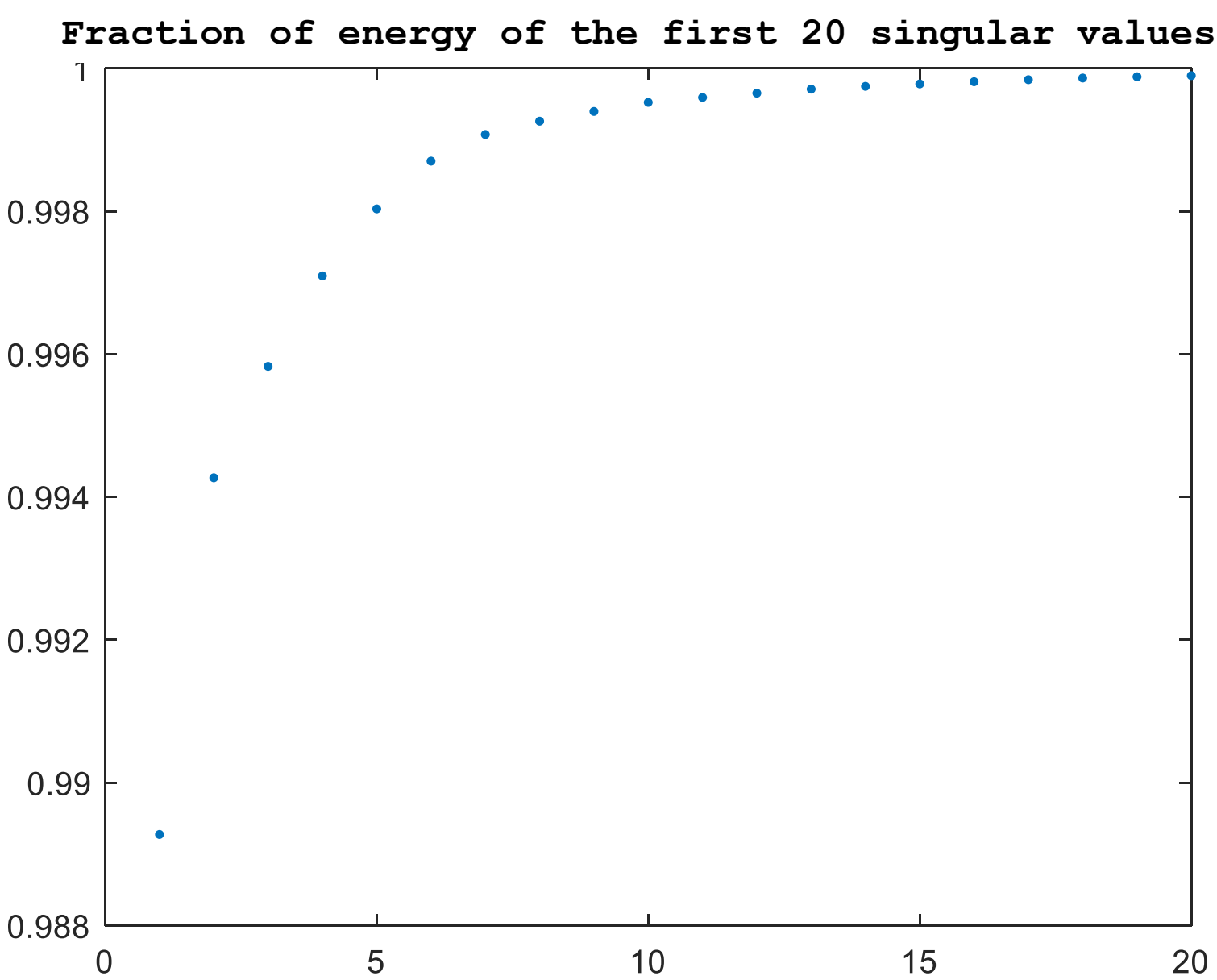
Nelle immagini n'energia è la somma dei quadrati dei valroi singolari quindi nelle immagini è la norma di forbeniusm al quadrato.

```
tau = cumsum(sigma.^2)/sum(sigma.^2);

plot(tau(1:20),'.','markersize',8)

title('Fraction of energy of the
first 20 singular values')
```

energy = sum of the squares of singular values


Fraction of energy of the first 20 singular values
1
0.998
0.996
0.994
0.992
0.99
0.988
0
5
10
15
20

Ora vediamo in bioinformatica cosa si fa cioè analisi di dati genici. Si parte dalla matrice dei dati A, che raccoglie esperimenti, in particolare si fissa una caratteristica , cioè un certo gene e si vede se questo gene sono presentio e in che misura, in topo 100 persone A sarà formata in riga dagli esperimenti cioè le persone e sulle colonne le caratteristiche, vale anche mettere le cose in ordine inverso tra righe e colonne (come se fosse la trasposta). L'idea fondamentalmente è sempre la stessa cioè proiettare in uno spazio minore (di dim k) i nostri dati.
IN bio informatica è fonamentale la riduzione della dimensionalità proprio perchè le matrici di dati hanno dim grandi e spesso anche sparse. L'obiettivo è clusterizzare i geni che si comportano in maniera simile in modo tale da tirar fuori caratteristiche in comune tra gli osservati

SVD Factorization and gene data analysis

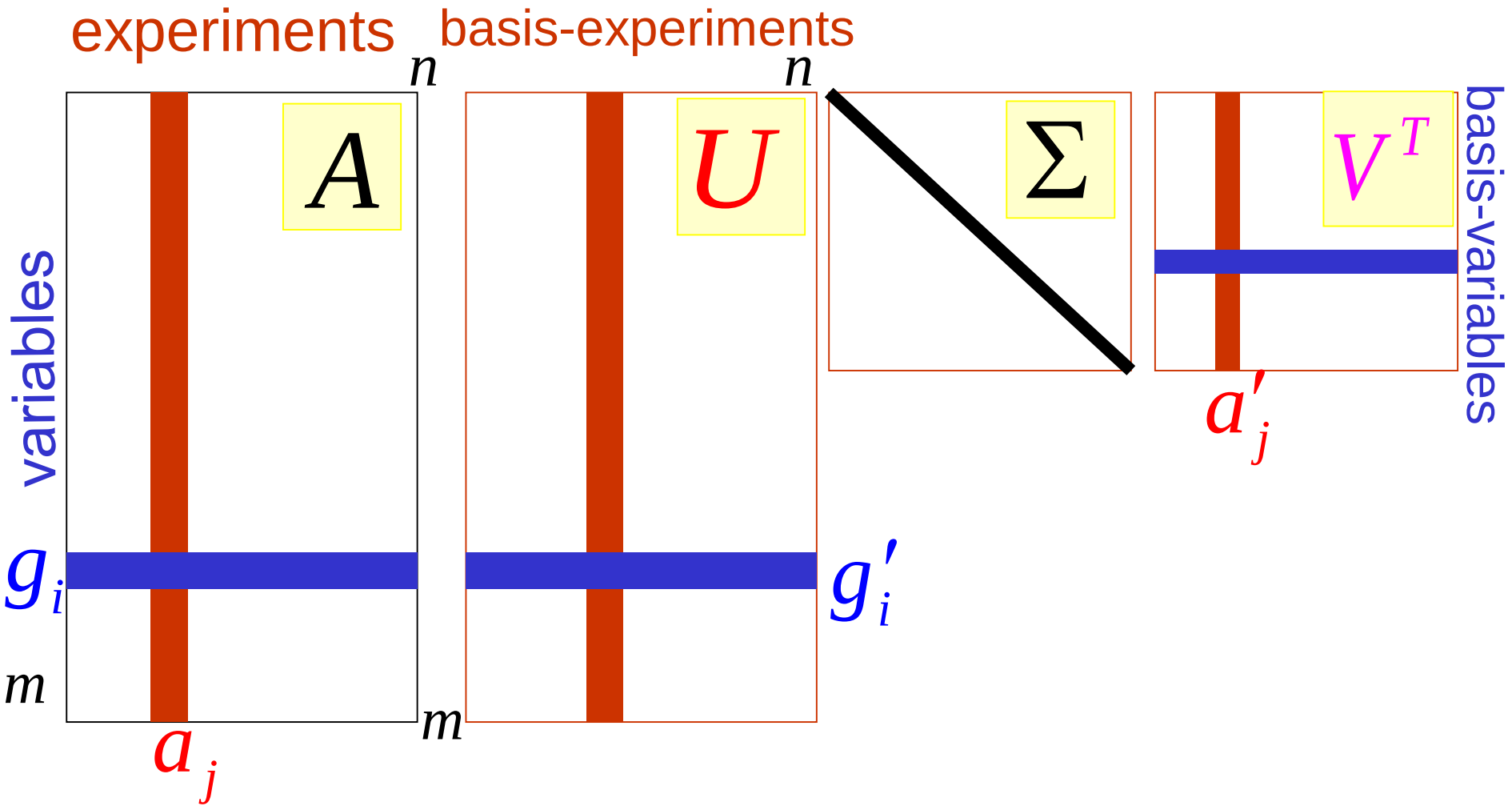
- ✓ SVD of A (data matrix)
- ✓ approximate A by means of a suitable $A^{(k)}$
- ✓ identification of correlations between the projections of the variables and of the experiments in the k-spaces of the left and right singular vectors of $A^{(k)}$

Bioinformatics – analysis of gene expression (Gene expression data analysis)

A set of genes is subjected to a battery of experiments. The goal is to cluster the genes that have a similar response and cluster the experiments that have a similar gene expression profile

Come al solito questa è l'espressione SVD di A che ha la rappresentazione come in foto: voglio esprimere Aj sulla base delle colonne di U e le componenti sono a'j in Vt. Sigma ha il fattore di scala, questo ovviamente vale poi anche per le righe gi

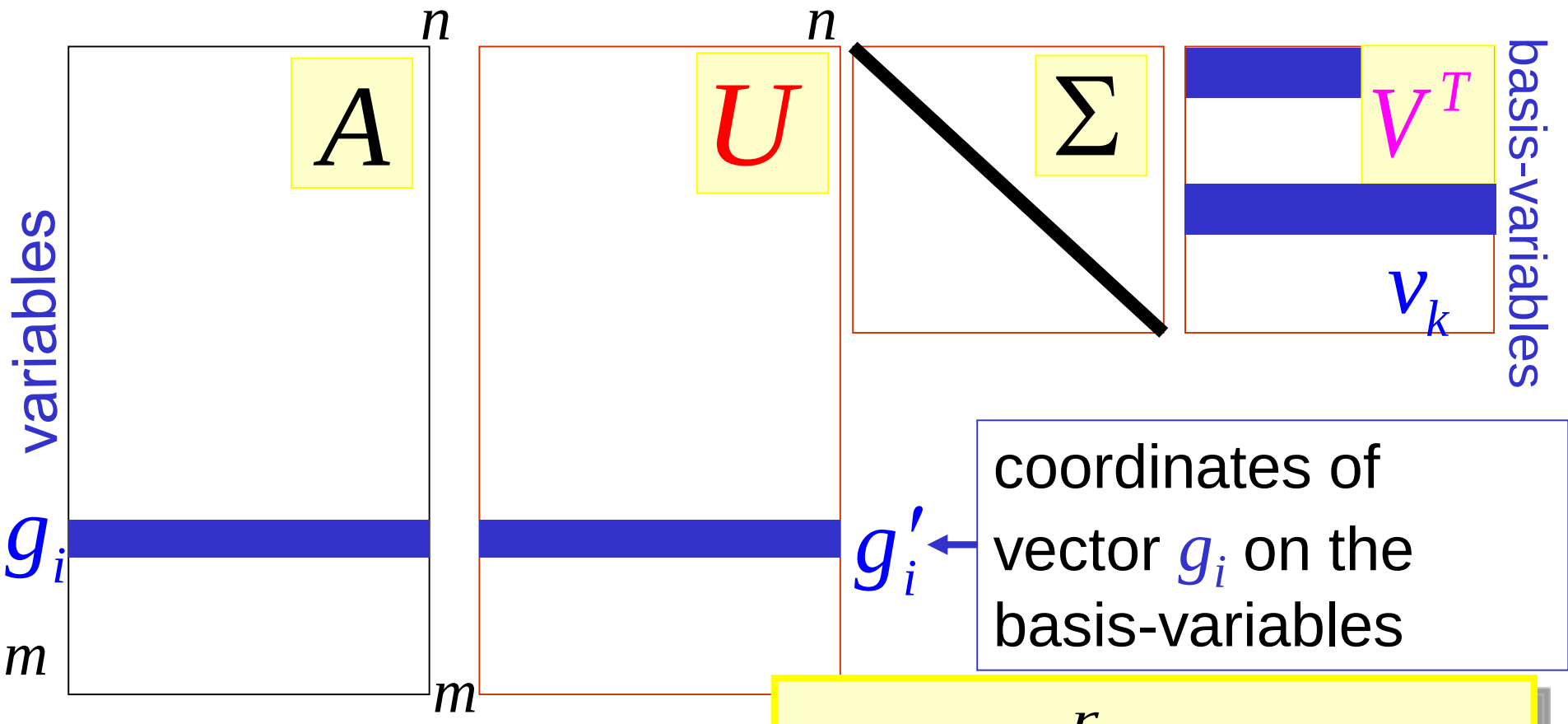
$$A = U \Sigma V^T$$



Qui ha invertito mettnedo gli esperimento sulle colonne e le var sulle righe ma l'idea è la stessa.

X es. se vogliamo esprimere la i-esimo gene gi lo rappresentiamo sulla base delle righe di Vt e le compoenti le troviamo nella i-esioma riga di U scalate per sigma.
Gi lo otteniamo come la sommatoria su base k fino a r della base Vk* le componenti ik scalate di sigma k

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



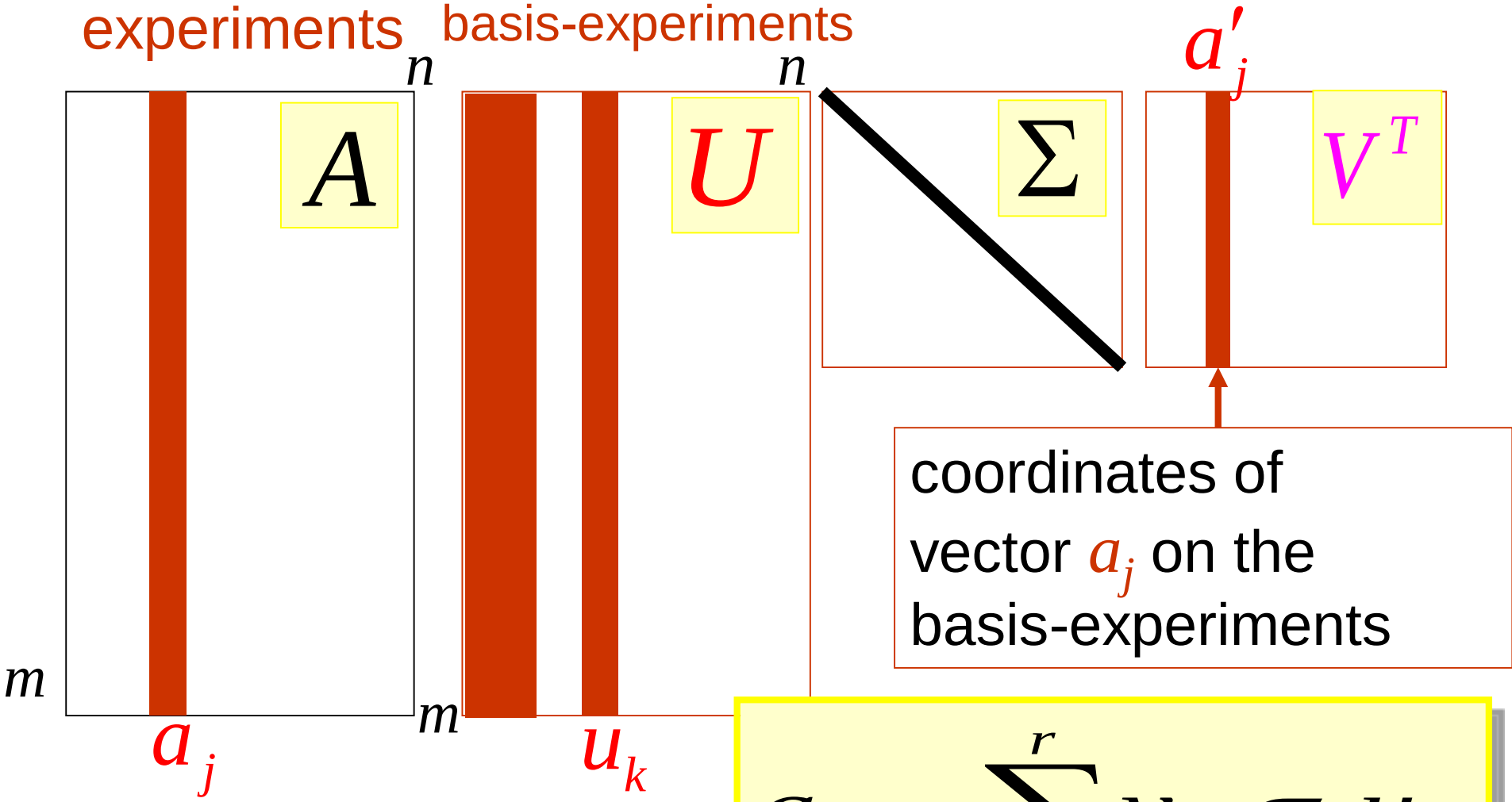
coordinates of vector g_i on the basis-variables

La barra sotto la variabile indica che sono vettori.

$$\underline{g}_i = \sum_{k=1}^{r} u_{ik} \sigma_k \underline{v}_k$$

Per aj invece abbiamo la sommatoria delle componenti vik scalate di sgima k * la base uk

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



$$\underline{a}_j = \sum_{k=1}^{r} v_{jk} \sigma_k \underline{u}_k$$

Questo è un esempio. Qui abbiamo 6 persone di cui sappiamo altezza e peso, vogliamo avere un unica caratteristica che sintentizza entrambe, cioè tipo è come se volessimo la taglia. con SVD ottengo una base ortogonale per il range di A che in questo caso è altezza e peso e considero solo il primo vettore di questa base

SVD and data dimensionality reduction

$$A = U_n \Sigma_n V^T$$

```
[Un,Sn,Vn]=svd(A,0)
```

```
Un =
   -0.3664    0.4781
   -0.4344   -0.1007
   -0.3882   -0.1897
   -0.3968    0.6043
   -0.4583   -0.5972
   -0.3985   -0.0599
```

height	weight
149	44
172	65
153	60
162	46
178	78
158	59

A

Poi otteniamo i valori singolari Sn che decrescono rapidamente perchè altezza e pes osono correlati se uno è alto probabilmete è anche più pesante. Vn sono le componenti. Come intrepretiamo quei valroi di Vn? Leggi le righe nella slide sotto a sx.

SVD and data dimensionality reduction

$$A = U_n \Sigma_n V^T$$

Sn =

423.2136	0
0	20.4514

Vn =

-397.5245	-0.9393	0.3430
-145.1622	-0.3430	-0.9393

7.0148

-19.2100

coord of **height** on the basis U_n

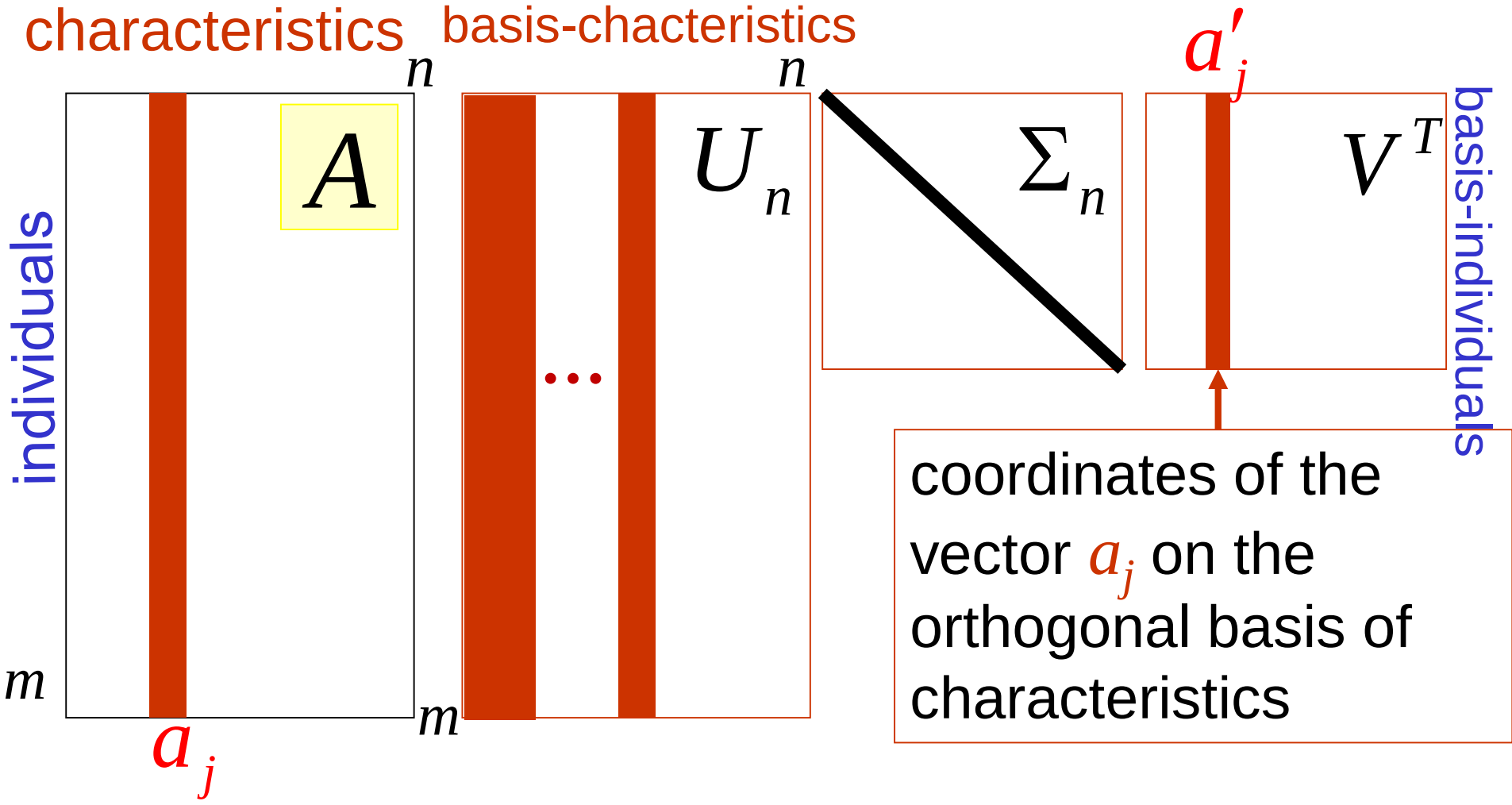
coord of **weight** on the basis U_n

height	weight
149	44
172	65
153	60
162	46
178	78
158	59

A

SVD and data dimensionality reduction

Skippa perchè è la stessa di sopra.



Possiamo costurire A a partire dalle matr di rango 1 quindi costruisco E1, quando lo faccio si vede che E1 ha un valore molto grande al primo posto e basso al secondo quindi già è un ottima ricostruzione di A,quindi in questo caso posso approssimare A usando E1. N.B Notare che siamo usando U1 cioè la prima colonna di u quindi stiamo dicendo che mi basta solo quella per rappresentarne le caratteristiche ,

$$A = U_n \Sigma_n V^T$$

```
>> sigma=diag(Sn)
sigma =
   423.2136
    20.4514
>> E1=sigma(1)*Un(:,1)*Vn(:,1)'
E1 =
   145.6463   53.1842
   172.7064   63.0655
   154.3308   56.3555
   157.7612   57.6081
   182.1890   66.5282
   158.4204   57.8488
```

$$E_1 = \sigma_1 u_1 v_1^T$$

height	weight
149	44
172	65
153	60
162	46
178	78
158	59

Faccio il prod tra Un e Sn (la diag di sigma) e ottengo una matrice dove ogni riga contiene le componenti delle persone sulla base delle colonne di V

$$A = U_n \Sigma_n V^T$$

```
>> newcomponents = Un*Sn

newcomponents =

  -155.0529    9.7774
  -183.8607   -2.0594
  -164.2983   -3.8799
  -167.9502   12.3578
  -193.9558  -12.2127
  -168.6520   -1.2255
```

each row contains the components of a person on the basis (columns of) V

height	weight
149	44
172	65
153	60
162	46
178	78
158	59

Moltiplico come detto in due slide fa, la prima componente di sigma per tutta la prima colonna di U1 quindi ottengo u1s che è il primo vettore della base scalato che possiamo definirlo come un indicatore della taglia. lo pone a -uls per togliere i negativi.
Size è la prima colonna della matrice UnSigman

$$A = U_n \Sigma_n V^T$$

```
>> u1s=sigma(1)*Un(:,1)
u1s =
   -155.0529
   -183.8607
   -164.2983
   -167.9502
   -193.9558
   -168.6520
```

$$\sigma_1 u_1$$

first scaled basis vector :
we can define it as an indicator of the "**size**" of a person

```
>> sizep =-u1s;
```

Pr riscotruire la prima colonna di A cioè l'altezza devo prendere il vettore -u1s che è la taglia e moltiplico per la prima componente di Vn, per approssimare il peso devo usare la seconda componente di Vn.
Quindi in generale la taglia mi consente di approssimare altezza e peso usando le componenti scalate di altezza e beso sulla base costituita dalla prima colonna della matrice U.

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

```
>> sizep =-u1s;
```

$$\sigma_1 u_1$$

first scaled basis vector : the “**size**”

```
-sizep*Vn(1,1)
```

approximation of **height** using **size**

```
-sizep*Vn(2,1)
```

approximation of **weight** using **size**

Questa è una soluzione di riduzione della dimensionalità

QUESTO è UN ESEMPIO CHE NON TRATTA NELLA LEZIONE IN ITA, DA VEDERE SE LO TRATTA IN QUEEL IN INGLESE

SVD and data analysis

We have a dataset (matrix D) with m=6356 rock samples and n=8 chemical elements. The first column of D is the amount (weight percent) of the first element in all the samples, the second column of D is the amount of the second element in all the samples, and so on

First 5 rows of D

51.9700	1.2500	14.2800	11.5700	7.0200	11.6700	2.1200	0.0700
50.2100	1.4600	16.4100	10.3900	7.4600	11.2700	2.9400	0.0700
50.0800	1.9300	15.6000	11.6200	7.6600	10.6900	2.9200	0.3400
51.0400	1.3500	16.4000	9.6900	7.2900	10.8200	2.6500	0.1300
52.2900	0.7400	15.0600	8.9700	8.1400	13.1900	1.8100	0.0400

SiO_2	TiO_2	Al_2O3	FeO-total	MgO	CaO	Na_2O	K_2O
Silicon dioxide	Titanium dioxide	Aluminium oxide	Iron oxide	Magnesium oxide	Calcium oxide (quicklime)	Sodium oxide	Potassium oxide

SVD and data analysis

We have a dataset (matrix D) with m=6356 rock samples and n=8 chemical elements

$$D = U_n \Sigma_n V^T$$

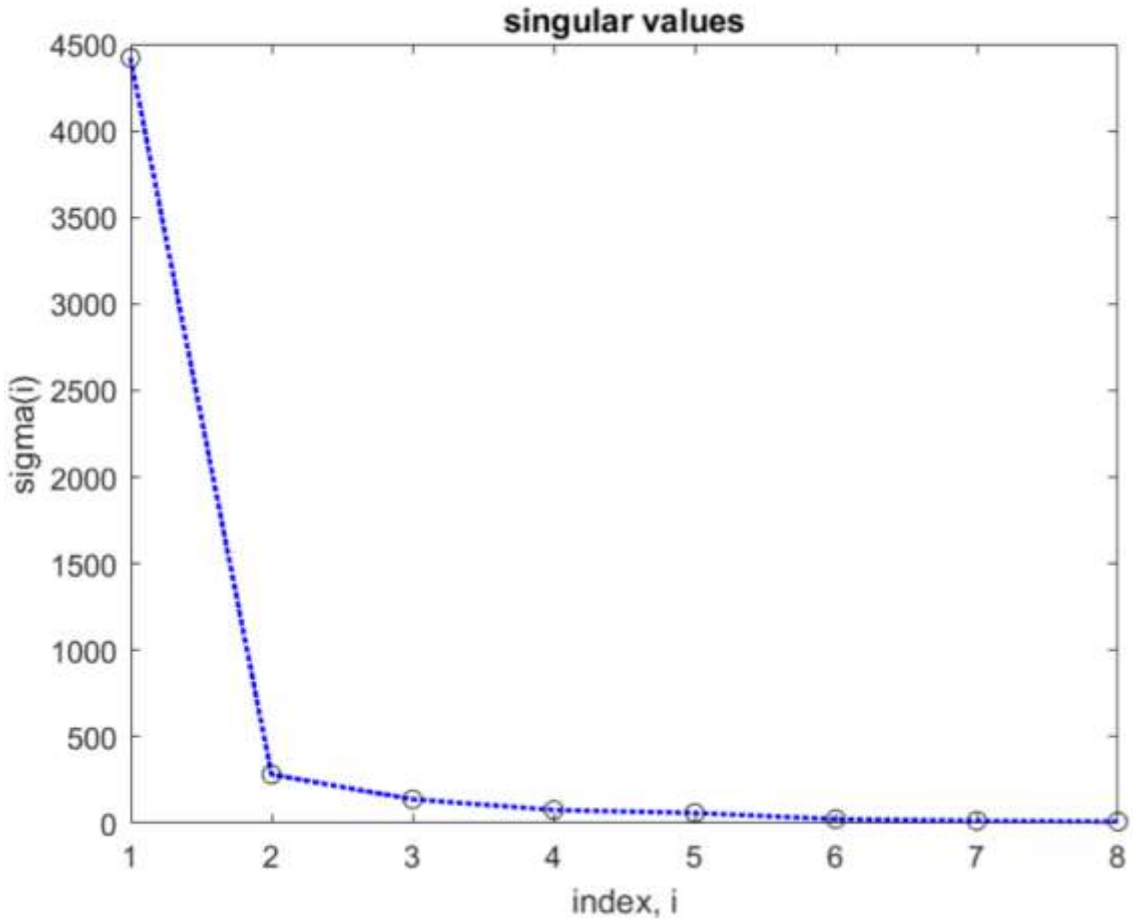
First 5 columns of V

SiO_2	-0.9088	0.0077	-0.1617	0.2098	0.3095
TiO_2	-0.0246	-0.0375	-0.1263	0.1514	-0.1005
Al_2O3	-0.2752	-0.3016	0.5678	0.1760	-0.6701
FeO-total	-0.1779	-0.0184	-0.6592	-0.4275	-0.5852
MgO	-0.1413	0.9232	0.2557	-0.1186	-0.1952
CaO	-0.2100	-0.2269	0.3657	-0.7800	0.2080
Na_2O	-0.0446	-0.0585	-0.0417	0.3024	-0.1453
K_2O	-0.0034	-0.0072	-0.0065	0.0734	0.0150

SVD and data analysis

$$D = U_n \Sigma_n V^T$$

We have a dataset (matrix D) with m=6356 rock samples and n=8 chemical elements



SVD and data analysis

$$D = U_n \Sigma_n V^T$$

column 1 of V

SiO2 -0.9088
TiO2 -0.0246
Al2O3 -0.2752
FeO_tot -0.1779
MgO -0.1413
CaO -0.2100
Na2O -0.0446
K2O -0.0034

column 2

SiO2 0.0077
TiO2 -0.0375
Al2O3 -0.3016
FeO_tot -0.0184
MgO 0.9232
CaO -0.2269
Na2O -0.0585
K2O -0.0072

column 3

SiO2 -0.1617
TiO2 -0.1263
Al2O3 0.5678
FeO_tot -0.6592
MgO 0.2557
CaO 0.3657
Na2O -0.0417
K2O -0.0065

column 4

SiO2 0.2098
TiO2 0.1514
Al2O3 0.1760
FeO_tot -0.4275
MgO -0.1186
CaO -0.7800
Na2O 0.3024
K2O 0.0734

column 5

SiO2 0.3095
TiO2 -0.1005
Al2O3 -0.6701
FeO_total -0.5852
MgO -0.1952
CaO 0.2080
Na2O -0.1453
K2O 0.0150

First 5 columns of V

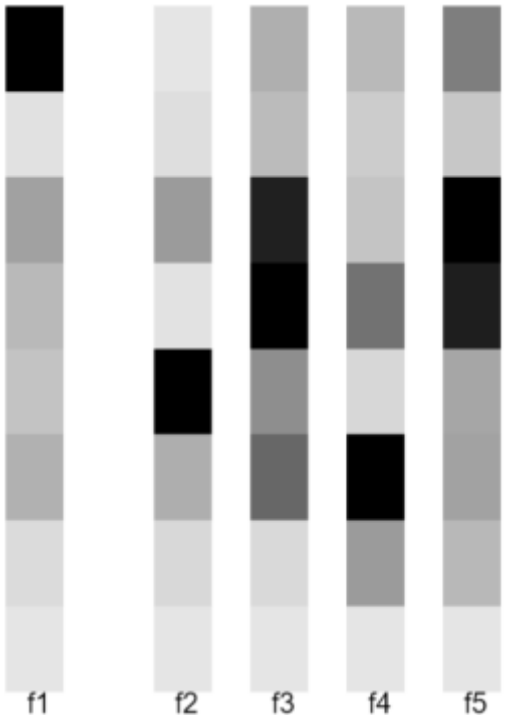
SVD and data analysis

$$D = U_n \Sigma_n V^T$$

Remember that the columns of V are a basis for the samples and this means that we can express any sample as a combination (mixture) of the columns of V. We can interpret the first column as a kind of **best representative of the set of samples**, the second column as a main correction (to the estimate in the first column) for capturing the variability of the samples and so on.

all rock samples contain a large amount of the first column—the typical sample. Only the first 5 columns are needed to describe the samples. The variability about the typical sample requires only four columns (2:5).

We can say that using the columns of V we can express **patterns of variability among chemical elements in the samples**

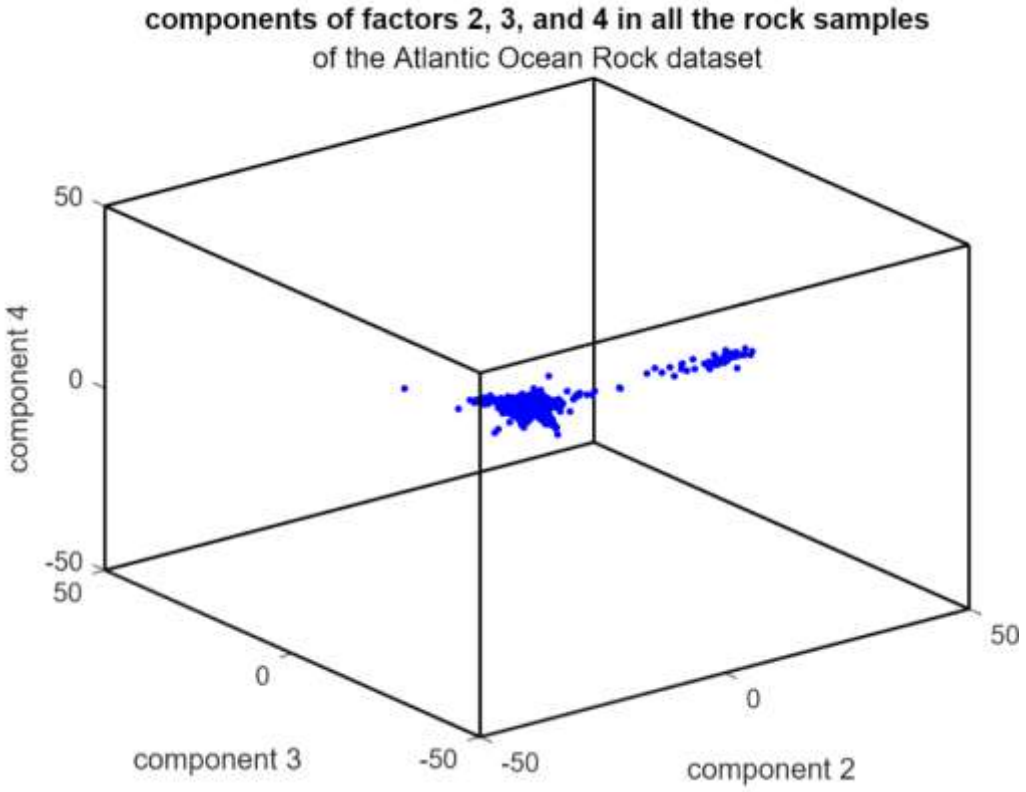


SVD and data analysis

$$D = U_n \Sigma_n V^T$$

The rows of $U\Sigma$ contain the components of the samples on the reference system of the columns of V, that is the first row contains the the weights of the combination (mixture) of the columns that gives the first sample, the second row contains the the weights of the combination of the columns that gives the second sample, and so on.

Note that the samples appear to form two clusters, one in which the variability is due to second column and the other due to third column. Note that the fourth components are almost equal for all samples



SVD and data analysis

$$D = U_n \Sigma_n V^T$$

Chemical elements = a linear combination (mixture) of the columns of U that make up a basis of the column space of D (space of chemical elements).

Working in the **space of chemical elements (column space of D) we are able to quantify patterns of variability among samples**

The sampling in our rock dataset is geographical, so the analysis of the range of D (columns of U) expresses **patterns of spatial variability** and can be used to detect **geographical clustering**

SVD and data analysis

$$D = U_n \Sigma_n V^T$$

Chemical elements = a linear combination (mixture) of the columns of U that make up a basis of the column space of D (space of chemical elements).

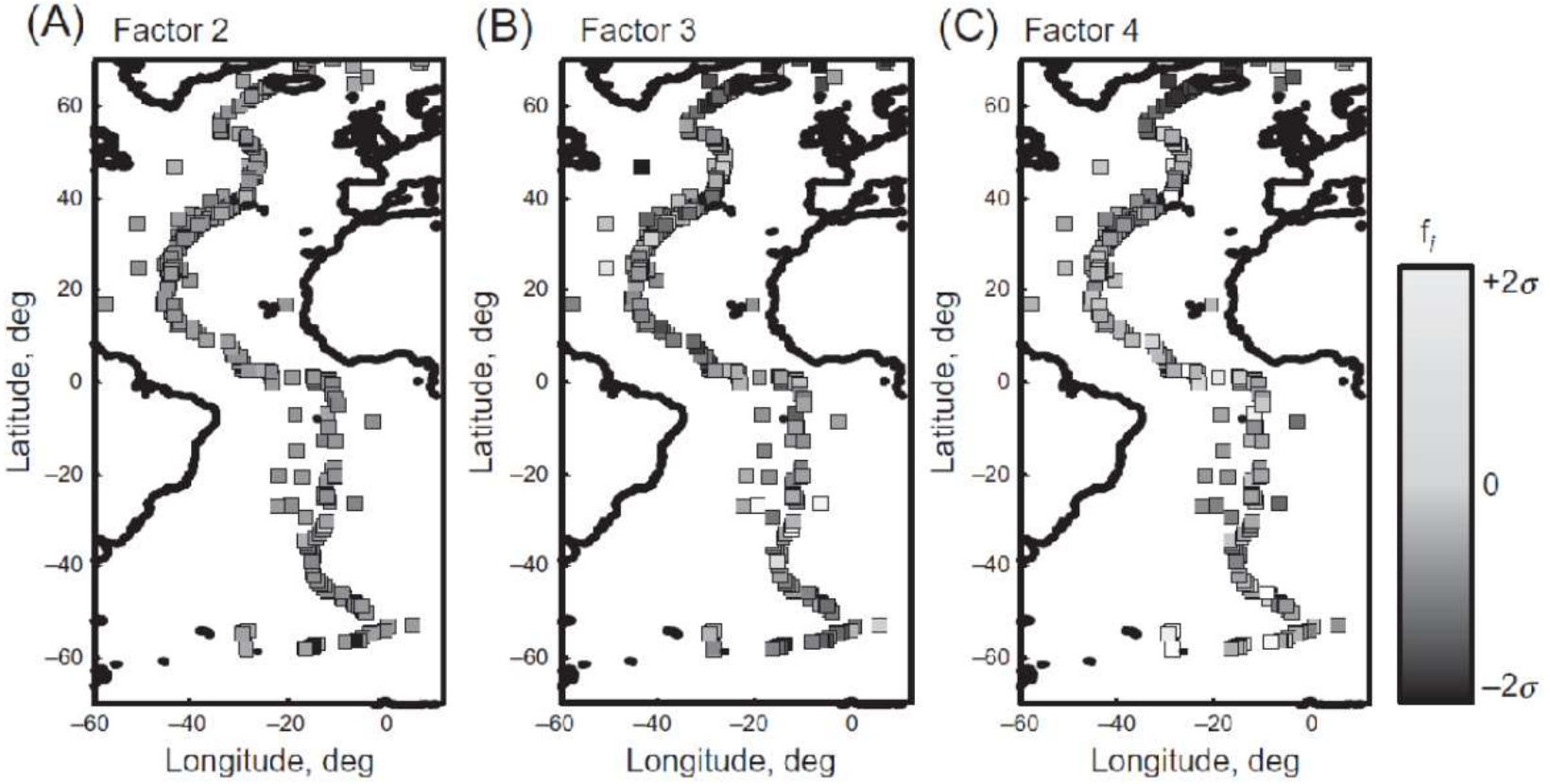
Working in the **space of chemical elements (column space of D) we are able to quantify patterns of variability among samples**

These spatial patterns can be brought out by plotting the components of each column of U on a map, with the i-th component plotted at the location of sample i and with its numerical value depicted by the color (or grey shade) of the square symbol

SVD and data analysis

$$D = U_n \Sigma_n V^T$$

We can see that the chemistry of the mid-Atlantic ridge (north-south bands of symbols) is strongly segmented, with nearly 10 degree long sections of fairly uniform chemistry punctuated by spatially-sharp jumps

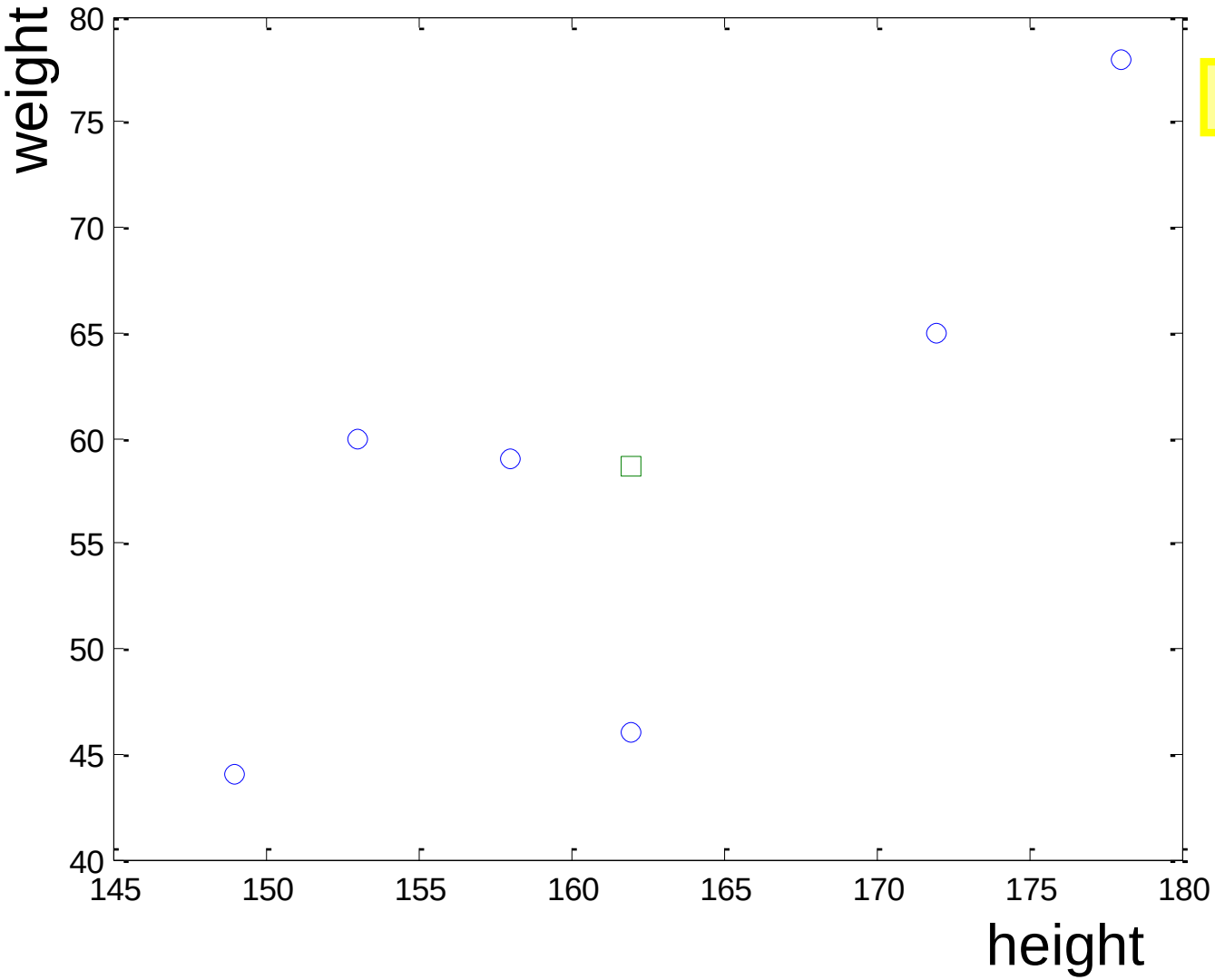


https://web.microsoftstream.com/video/2b8ecd39-0c13-4b7a-9082-3c10e9782517

SVD Factorization and PCA (Principal Component Analysis)

PCA is an orthogonal linear transformation of the data that maps the data into a **new reference system** in order to **maximize the variance** (with respect to any other projection) associated with the first coordinate (**first principal component**), and then that associated with the second coordinate (**second principal component**), and so on

SVD Factorization and PCA (Principal Component Analysis)



height	weight
149	44
172	65
153	60
162	46
178	78
158	59

data centering (for each column, the average value of a column is subtracted from that column)

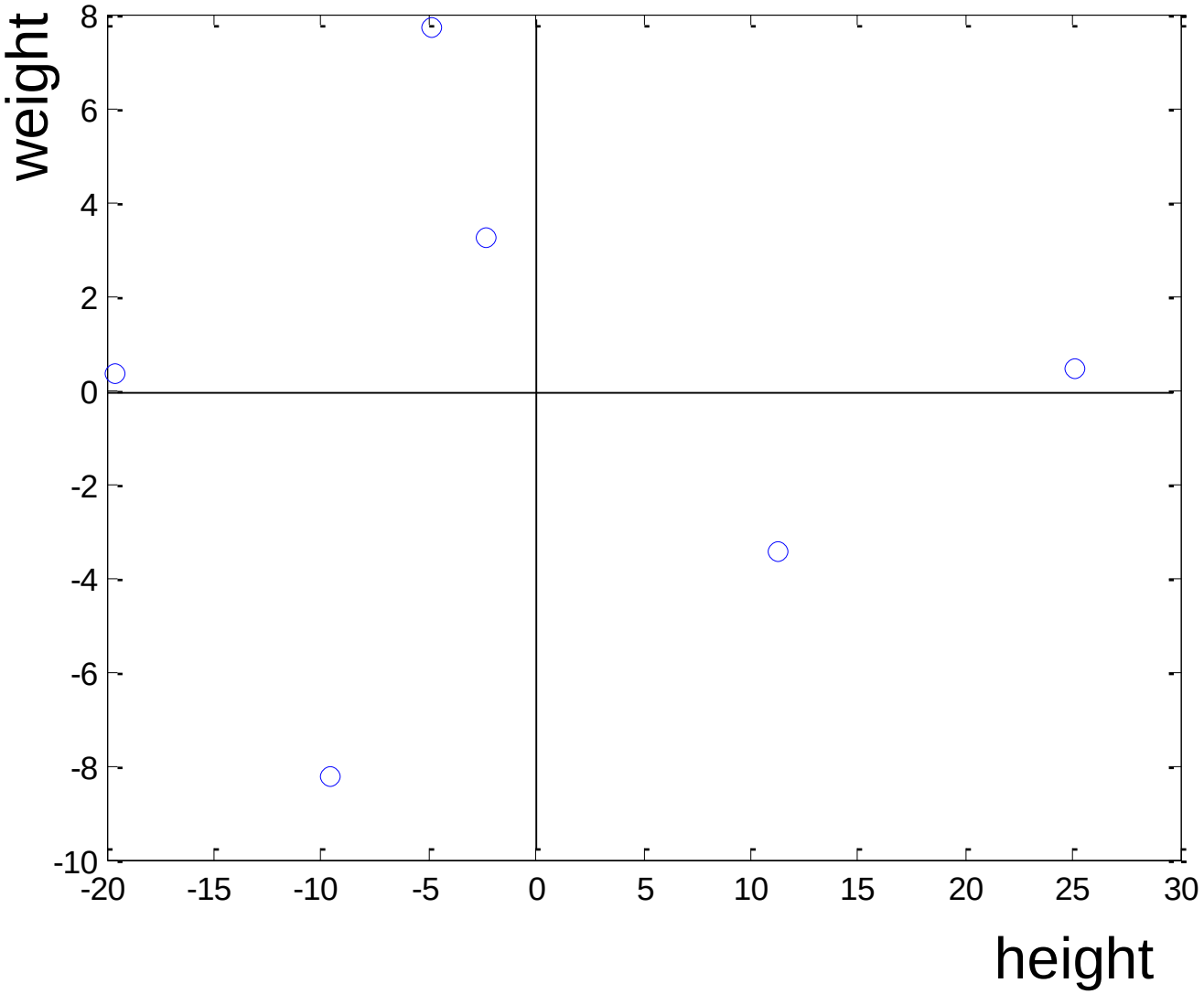
```
>> X=A-ones(size(A,1),1)*mean(A)
X =
  -13.0000  -14.6667
   10.0000    6.3333
   -9.0000    1.3333
         0  -12.6667
   16.0000   19.3333
   -4.0000    0.3333
```

$X^T X$

covariance matrix
(height and weight)

$$c_{hw} = \frac{1}{n}\sum_{i=1,}^{n}\left(h_i - mean(height)\right)\left(w_i - mean(weight)\right)$$

data centering (for each column, the average value of a column is subtracted from that column)



PCA

1. Compute **eigenvectors** and **eigenvalues** of the **covariance** matrix (of **centered** data)
2. The **first** eigenvector if the **first** principal component
3. The **first** eigenvalue is the **variance** associated to the **first** principal component
4. The second eigenvector ….
5. The second eigenvalue …..
6. …..

$$X^T XP = P\Lambda$$

PCA is named differently in some disciplines

1. Karhunen – Loeve transform (KLT), signal processing
2. Hotelling transform, statistics, control
3. Proper orthogonal decomposition (POD), mechanical engineering
4. Empirical orthogonal functions (EOF), meteorology
5. Empirical modal analysis, structural engineering…..
6. …..

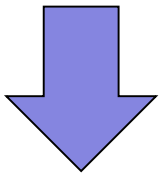
PCA is **equivalent** to SVD

PCA

$$X^T X P = P\Lambda$$

SVD

$$X = U_n \Sigma_n V^T$$



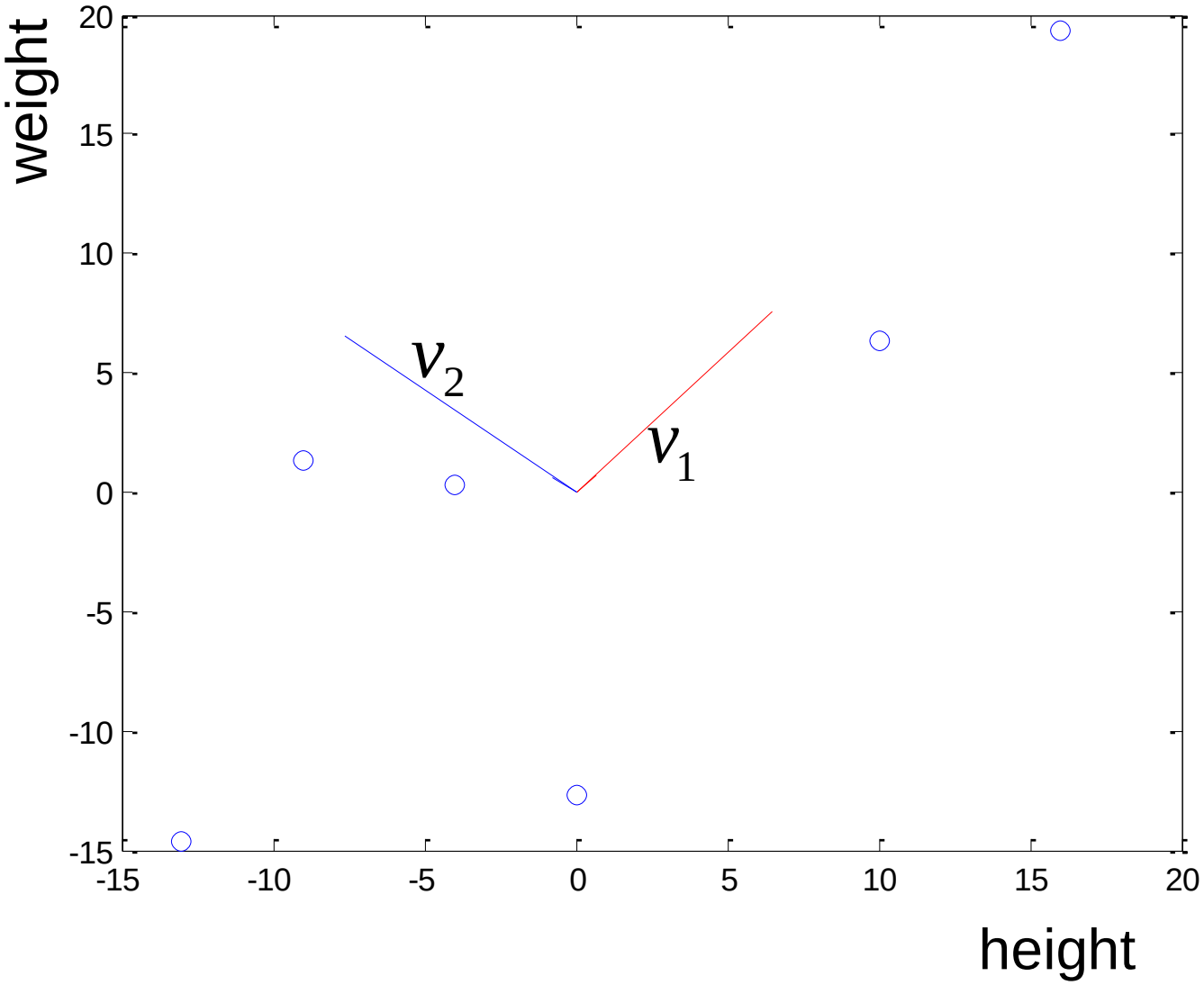
$$P = V$$

$$\lambda_i = \sigma_i^2$$

```
>> [Un,Sn,V]=svd(X,0)
Un =
   -0.5514    0.0259
    0.3184   -0.2828
   -0.1364    0.6282
   -0.2705   -0.6730
    0.7060    0.0362
   -0.0662    0.2654
Sn =
   35.5408         0
         0   12.2551
V =
    0.6511   -0.7590
    0.7590    0.6511
```

first principal component	second principal component

```
>> hold on
>> VV=10*V;
>> plot([0 VV(1,1)],[0 VV(2,1)],'r',[0 VV(1,2)],[0 VV(2,2)],'b')
```

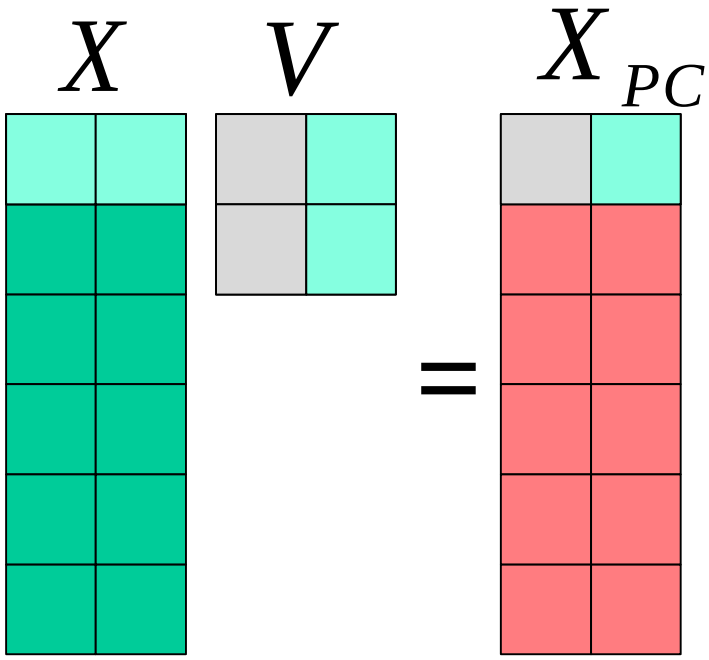


data expressed in the basis of principal components (columns of V)

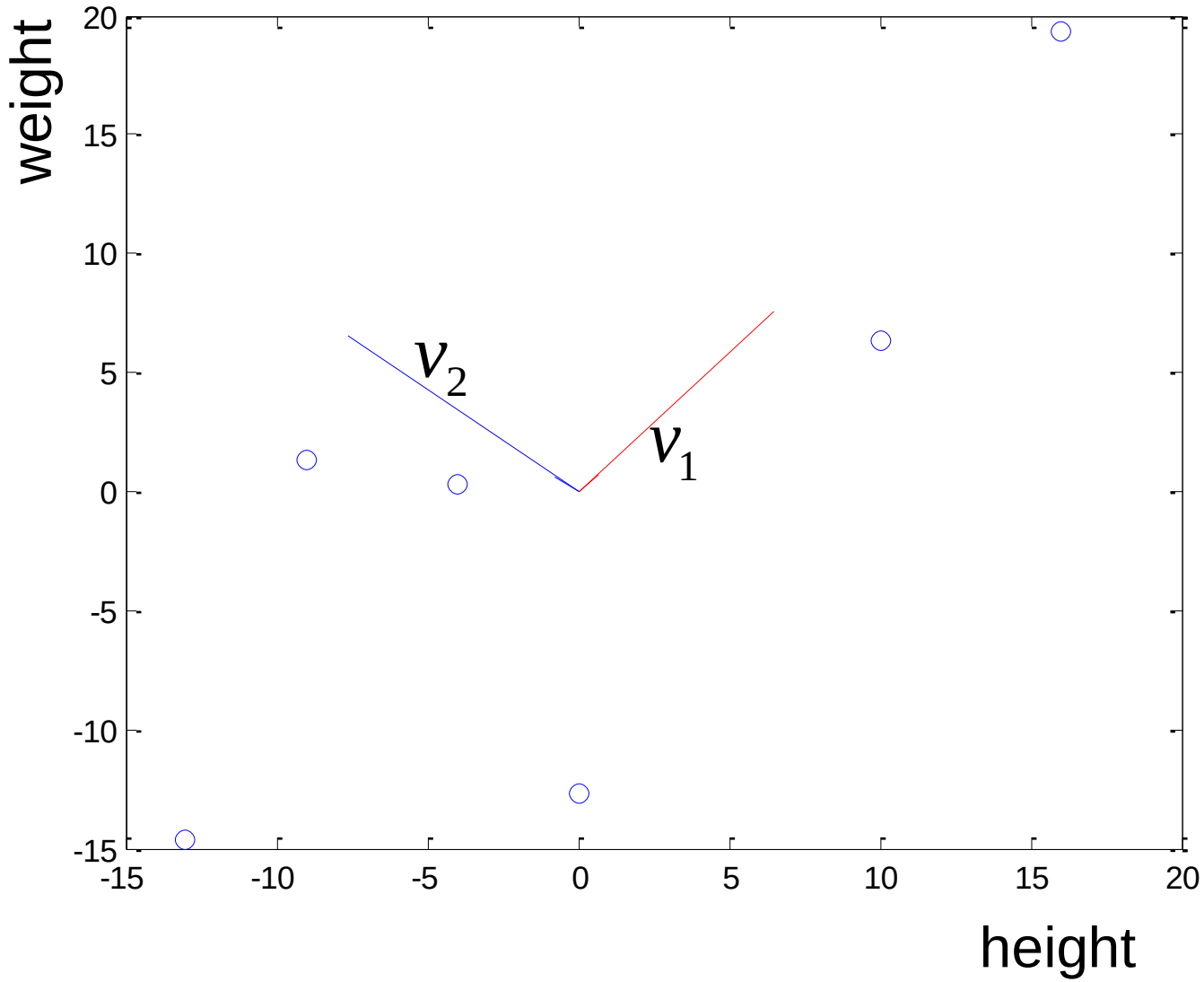
```
>> Xpc=X*V
Xpc =
  -19.5962    0.3175
   11.3179   -3.4663
   -4.8479    7.6991
   -9.6139   -8.2472
   25.0914    0.4439
   -2.3514    3.2530
>> Un*Sn
ans =
  -19.5962    0.3175
   11.3179   -3.4663
   -4.8479    7.6991
   -9.6139   -8.2472
   25.0914    0.4439
   -2.3514    3.2530
```

$$X = U_n \Sigma_n V^T$$

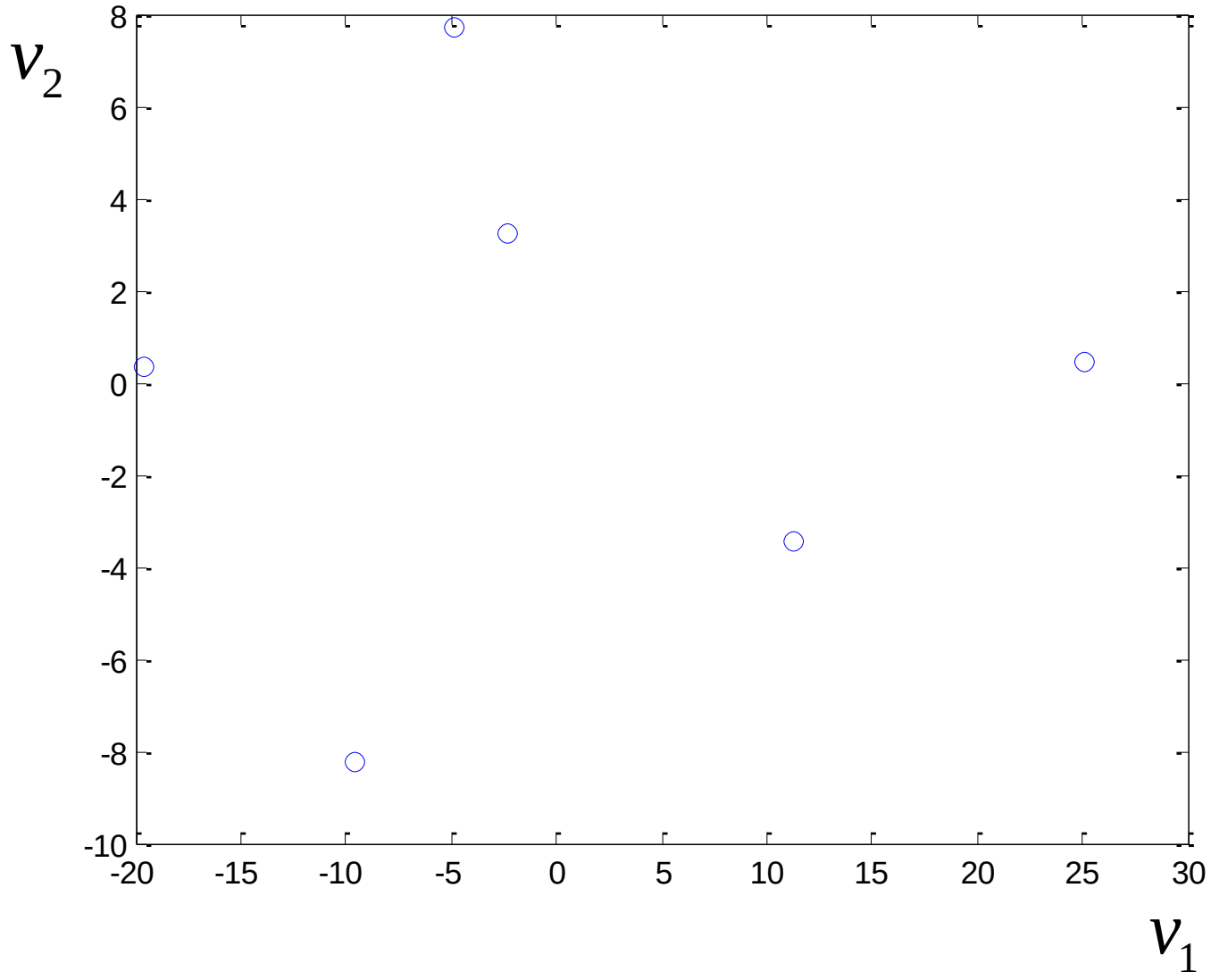
$$XV = U_n \Sigma_n$$



v_1 is the direction of greatest variability of data (maximum of the variance)



data expressed in the basis of principal components (columns of V)



```
>> std(X)
ans =
   11.1535   12.5804

>> std(Xpc)
ans =
   15.8943    5.4807

% compute standard deviation
% of the data projected on
% other orthogonal bases
>> teta=linspace(0,2*pi,20);
>> for i=1:20
     R=[cos(teta(i))
      sin(teta(i));
      -sin(teta(i))
      cos(teta(i))];
      DevStandard(i,:)=
            std(X*R);
end
```

```
>> DevStandard
DevStandard =
   11.1535   12.5804
    7.7722   14.9084
    5.5265   15.8784
    6.8848   15.3384
   10.1917   13.3714
   13.2718   10.3211
   15.2883    6.9954
   15.8857    5.5055
   14.9720    7.6490
   12.6899   11.0287
    9.4658   13.8948
    6.3232   15.5783
    5.7539   15.7974
    8.4778   14.5188
   11.8292   11.9472
   14.4423    8.6074
   15.7756    5.8135
   15.6144    6.2335
   13.9833    9.3345
   11.1535   12.5804
```