Richiamo su autoivalore e autovettore. La matrice su cui operiamo deve essere quadrata, non esistono autovalori e autovetotri di matrici non quadrate. Si dice che lambda è autovalore di A se Ax è = malbdax con x != dal vetotre nullo. Lamda è l'autovalore e x è il corrispondente autovettore.

Review of eigenvalues and eigenvectors

A is a **square** matrix of order n

corresponding eigenvector

eigenvalue

$$A\boxed{x} = \boxed{\lambda}\boxed{x} \ , \quad x \neq 0$$

questa condizione è necessaria in quanto sicuramente con il vettore nullo quella condzione è sempre vera.

there are n eigenvalues (not necessarily distinct)

per una matrice di ordine n ci saranno n autovalori non necessariamente distinti. Quindi possiamo scrivere la condizione come sta qui sotto in giallo.

$$Ax_i = \lambda_i x_i \ , \quad i = 1:n$$

lambda è un numero reale ocomplesso, x è un vettore di n componenti sia numeri reali che numeri complessi.

Review of eigenvalues and eigenvectors

A is a **square** matrix of order n

there are n eigenvalues (not necessarily distinct)

$$Ax_i = \lambda_i x_i \;,\quad i = 1:n$$

In generale diciamo che se gli n autovalori sono linearmente indipendenti possiamo scrivere quello in giallo con quello in verde (che è un modo per sintetizzare queste n relazioni) dove:

$$AX = X\Lambda$$

1 2 3

X is the matrix of **eigenvectors** (columns)

gli autovettori sono le colonne

Λ is the diagonal matrix of **eigenvalues**

lamda 'è la matrice diagonale degli autovalori. gli autovalori stanno sulla diagonale, il resto è 0.

Review of eigenvalues and eigenvectors

Questo comando genera gli autovalori

```
eig(A)
```

Chiamato così generà X che ha gli autovettori e lamda che è la matrice degli autovalori.

```
[X,Lambda]=eig(A)
```

Gli autovalori non sono necessariamente distinti, possono essere numri complessi

- ✓ **eigenvalues** can be complex numbers
- ✓ **eigenvectors** can have complex components
- ✓ **eigenvectors** can be arbitrarily scaled

Gli autovettori possono avere componenti complesse e possono essere scalati arbitrariamente.

Se la matrice degli autovettori è linearmenter indipendete (X non dev essere singolari) allora esiste l'inversa e io moltiplico ambo i membri (quelli della formula vista due slide prima) per l'inversa cioè da AX = Xlamda e moltiòicando ambo i membri con X alla meno 1 allora otteniamo la forma che vediamo nella slide attuale

Review of eigenvalues and eigenvectors

Questa formula si chiama decomposizione spettrale.

$$A = X \Lambda X^{-1}$$

eigenvector (spectral) decomposition

only if A is non defective ($A^T A = A A^T$)

the **eigenvectors** (columns of X) are a **basis** of R^n

important property:

$$A^p = X \Lambda^p X^{-1}$$

Se eleviamo A alla p abbiamo questa relazione icoè gli autovettore sono gli stessi e gli autovalori sono anche loro elevati alla p.

Una cosa interessante è che se io prendo una qualsiasi matrice T non singolare e trasformo A in quel modo (cioè molt a sx per inversa di T e dx per T) ottengono una matrice B che ha gli stessi autovalori di A

Review of eigenvalues and eigenvectors

Questa si chiama quindi trasformazione di similitudine che quindi preserva gli autovalori ed è spesso usata per determinare il calcolo degli autovalori.

T **similarity transformation**

$$B = T^{-1} A T$$

B and A have the same **eigenvalues**

similarity transformations preserve **eigenvalues**

Questa dovrebbe essere la dimostrazione che non mostra a lezione, la dimostrazione parte da AX = Xlambda pone x a quella roba e usa la decomposizione spettrale cioè moltiplica ambo i membri per T ^ -1, poi il resto è da vedere.

Review of eigenvalues and eigenvectors

T similarity transformation

$$B = T^{-1}AT$$

B and A have the same **eigenvalues**

$$AX = X\boxed{\Lambda}$$

we set $X = TY$

$$ATY = TY\Lambda \qquad T^{-1}ATY = T^{-1}TY\Lambda$$

$$BY = Y\boxed{\Lambda} \qquad \boxed{Y = T^{-1}X}$$

Se A è simmetrica cioè A = At alllora gli autovettori sono reali e ortogonali e gli autovalori sono reali.

Review of eigenvalues and eigenvectors

A **symmetric matrix**

$$A = A^T$$

eigenvalues are **real**

eigenvectors are **real** and **orthogonal**

In questo caso A sarà uguale a questa formula qui dove Q è una matrice ortogoonale che è la ortogonal spectral decomposition

$$A = Q\Lambda Q^T$$

orthogonal spectral decomposition

N.Bla decomposizione di una matrice è la scomposizione di una matrice in un prodotto di più matrici (decomposizione = fattorizzazione) e la sua utilità è che alcune operazioni complesse on le matrici non possono essere risolte il modo efficiente per cui scomponendo riduciamo una matrice in parti costituitve che rendono più semplice le operazioni

Nel caso in cui abbiamo che A è triangolare (o diagonale che è un caso particolare di matr triangolare) allora gli elementi della diagonale coincidono con gli autovalori. Cioè non devo fare nessun calcolo pe trovare gli autovalori.

Review of eigenvalues and eigenvectors

A **triangular matrix**

A **diagonal matrix**

⇓

eigenvalues = diagonal entries of A

L'idea infatti del agl per il calcolo degli autovalori è di applicare una serie di trasformazioni di similituidini in modo tale da ottenre una matrice triangolare superiore e sfruttare quindi questa proprietà.

Il calcolo degli autovalori è un problema ben condizionato quindi se modifico di poco gli elemtni della matrice non cambio di molto i suoi autovalori, non è proprio vero, però il condizionamneto non dipende dagli elementi di A piuttosto da come è fatta la matrice degli autovettori cioè dall'indicie di condiziomento k(x) della matrice degli autovettori.

Review of eigenvalues and eigenvectors

conditioning (sensibility of **eigenvalues** to perturbations of entries of the matrix)

↓

it is estimated by the condition number of the matrix of **eigenvectors**

$$\kappa(X)$$

$$\|\Delta\Lambda\| \le \kappa(X)\|\Delta A\|$$

l'errore in norma sulla perturbazione che ottengo sugli autovalori se perturbo la matrice a con Delta A dipende dall'indice di condizionamento di X.

Il metodo delle potenze sfrutta le proprietà elencate prima e permette di calcolare l' autovettore corrispondendie all'uatovalore di massimo modulo, cioè formalmente risolve il blocco verde (la formula), in realtà con un piccolo accorgimento può calcolare anche l'autovalore di massimo modulo. (nella catena di markov non ci interessa perché già sapevamo che era 1)

power method

$$A x_{\max} = \lambda_{\max} x_{\max}$$

compute the eigenvector corresponding to the eigenvalue of maximum modulus

è un metodo iterativo dvoe si parte da un vettore arbitrario che ad ogni passo viene si aggiorna moltiplicando A per il vettore al passo precedente (metodo di punto fisso) generando quindi l'approssimazione al passo successivo. Dopo p passo w tende all'auto vettore corrispondente all'autovalore di massimo modulo.

iterative method

w_0 arbitrary vector

wp tende a x max per p che tende all'infinito

$$w_p = A w_{p-1}$$

$$w_p \to x_{\max}$$

$$p \to \infty$$

$$w_p = A^p w_0$$

qui semplicemente dice che dopo p passdi wp = a quella roba li.

Ora lo dimostriamo. Ad ogni passo quello che facciamo è approssimare anche l'autovalore. (a partire dalla formula questa Axmax = lamda etc, se moltiplico ambo i membri per xmax trapsoto ottengo a sx xmaxT AxMax e a dx lambda max * prod salare tra xmac e xmaxT che è un numero, al passo successivo divido amb i membri per questo numero e ottengo a sx xmaxT*Axmax/prod scalare tra xmaxT*x = lambda)

power method

algorithm

$$A x_{max} = \lambda_{max} x_{max}$$

$$w_p = A w_{p-1}$$

Quindi con questo Rauleigh ratio posso calcolare l'autovalore corrispondente al passo p che all'ultimo passo corrisponde l'autovalore di massimo mo0dulo. Quindi con p tende a infitno lp tende a lambda max.

Rayleigh ratio

quoziente di rayleigh.

$$\ell_p = \frac{w_p^T A w_p}{w_p^T w_p}$$

$$\ell_p \to \lambda_{max}$$

$$p \to \infty$$

it also computes the **eigenvalue** **of maximum modulus**

N.B è anche possibile calcolare l'autovettore con noto autovalore

Per vitare che i vetotri diventino troppo grandi in modulo teniamo i vettori generati scalati in modo tale che la norma 2 sia = 1 in modo che l'autovettore espresso come versore della direzione, cioè appunto vettore con nomra 2 = 1. Zp è il versore di Wp. Calcoliamo in quel modo wp+1 usando quindi il versore e di conseguenza anche lp. (il quoziente di reily) userà il verosore.

power method

scaled version

algorithm

w_0 arbitrary vector

$$z_p = \frac{w_p}{\left\| w_p \right\|_2} \Rightarrow \left\| z_p \right\|_2 = 1$$

$$w_{p+1} = A z_p$$

$$z_p \to x_{\max} \; ; \; l_p \to \lambda_{\max}$$

$$p \to \infty$$

$$l_p = \frac{z_p^T A z_p}{z_p^T z_p} = z_p^T w_{p+1}$$

Dimostriamo la convergenza, supponiamo che A non difettiva e che abbia gli autovettori linearmente indipedneti, allora in questo caso formano una base, ciè quei xi che sono vettori li possiamo usare per espiremere qualsiasi vettore come ad esempio il vettore w0. (lo esprimo come una combinazione lineare della base costituita dagli autovettori di A,) alfa con i sono le componenti di w0 su quella base.

power method

convergence

$$Ax_i = \lambda_i x_i \quad , i = 1:n$$

Detto questo è chiaro che per la formula vista prima moltiplichiamo w0 con A quindi porto A dentro (sfrutto la linearità del prod matrice vettore). Ma Ax è scritto in alto che è = a lambdai xi quindi sostituisco. L'ultimo blocco generalizza la tenica per il passo p-esimo

$$w_0 = \sum_{i=1}^{n} \alpha_i x_i$$

representation of w_0 on the **eigenvectors** basis

$$Aw_0 = A\sum_{i=1}^{n} \alpha_i x_i$$

$$Aw_0 = \sum_{i=1}^{n} \alpha_i A x_i$$

$$Aw_0 = \sum_{i=1}^{n} \alpha_i \lambda_i x_i$$

$$A^p w_0 = \sum_{i=1}^{n} \alpha_i \lambda_i^p x_i$$

Ora supponiamo che il primo autovalore è il più grande e lo porto fuori dalla sommatoria e scrivo in quel modo Apw0.

power method

convergence

$$Ax_i = \lambda_i x_i \quad , i = 1:n$$

$$A^p w_0 = \sum_{i=1}^{n} \alpha_i \lambda_i^p x_i$$

suppose the first **eigenvalue** be the largest

$$A^p w_0 = \alpha_1 \lambda_1^p x_1 + \sum_{i=2}^{n} \alpha_i \lambda_i^p x_i$$

a questo punto metto in evidenza lambda1 alla p e ottengo questa formula.

$$A^p w_0 = \lambda_1^p \left(\alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^p x_i \right)$$

Al limite di p tendente all'infinito di questa roba accade la roba cerchiata tende a 0 quindi rimane solo il primo auiovalore che abbiamo detto essere il più grande.

Poichè quella roba tende a 0 con p tendente all'inifit allora rimane che il vettore Apw0 si orienta nella direzione di xi

power method

convergence

$$A x_i = \lambda_i x_i \quad , i = 1:n$$

suppose: the first eigenvalue is the largest

$$A^p w_0 = \boxed{\lambda_1^p \left(\alpha_1 x_1 \right.} + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^p x_i \left. \right)$$

$$A^p w_0 \to c x_1$$

$$p \to \infty$$

$$0$$

$$p \to \infty$$

La velocità di convergenza di questa roba dipende dal rapporto dei due più grandi autovalori, quindi il piuù piccolo è il rapporto più veloce corvege.

power method

convergence

$$A x_i = \lambda_i x_i \quad , i = 1:n$$

the first eigenvalue is the largest

$$A^p w_0 = \lambda_1^p \left(\alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^p x_i \right)$$

the rate of convergence depends on the ratio of the **two largest** eigenvalues, the smaller the ratio the faster the convergence

$$\frac{|\lambda_2|}{|\lambda_1|}$$

Spiega l'idea dell'algoritmo QR, questo alg si basa sulla applicazione iterativa della fattorizzazione QR , l'idea è di usare una catena di trasformazioni di similitudine ortogonali che trasfomrano a in un matrice triangolare.

QR algorithm for eigenvalues/vectors

$$AX = X\Lambda$$

the algorithm is based on the iterative application of QR factorization

idea: a sequence of orthogonal similarity transformations transforms A into a triangular matrix

Calcolo la fattorizzazione QR di A dopodichè non faccio esattamente questo ma piuttosto faccio il prodotto RQ che genera una matrice A1 quadrata (A anche A è quadrata) dello stesso ordine di A. Ora A1 e A hanno gli stessi autovalori perché sono legati da una trasf di similituidine, cioè RQ, ma RQ è una trasf di similitudine? Lo mostra moltiplicando ambo i membri di A=QR per Qtrasposto, Q * Qt è la matrice di indentità quindi ottengo QT*A = R.

$$A = QR \qquad \Rightarrow \qquad R = Q^T A$$

$$A_1 \equiv RQ \qquad \qquad A_1 = Q^T A Q$$

orthogonal similarity transformation

Ora prendo a = rq sostitiusco con la definizione di R e ottengo quindi quest'ultimo elemento. Questa è quindi una trasformazione di similitudine ortgonali quindi preserva gli autovlari e cconsente di calcolare semplicemente gli autovettori..

Andandio avanti A2 sarà la trasf di similitudine di A etc. Ma le matrici che determinano la trasformazione sono QR che quindi devono avere l'indice, ad ogni passo si usa una matrice QR diversa.

Questo è quello che ha detto prima. alla fine cioè gli indici su Q e R.

QR algorithm for eigenvalues/vectors

$$AX = X\Lambda$$

the algorithm is based on the iterative application of QR factorization

idea: a sequence of orthogonal similarity transformations transforms A into a triangular matrix

$$A = Q_0 R_0$$

for $i = 0, 1, \ldots$

$$A_{i+1} \equiv R_i Q_i$$

$$A = Q_0 R_0$$

$$A_1 \equiv R_0 Q_0 \qquad A_1 = Q_1 R_1$$

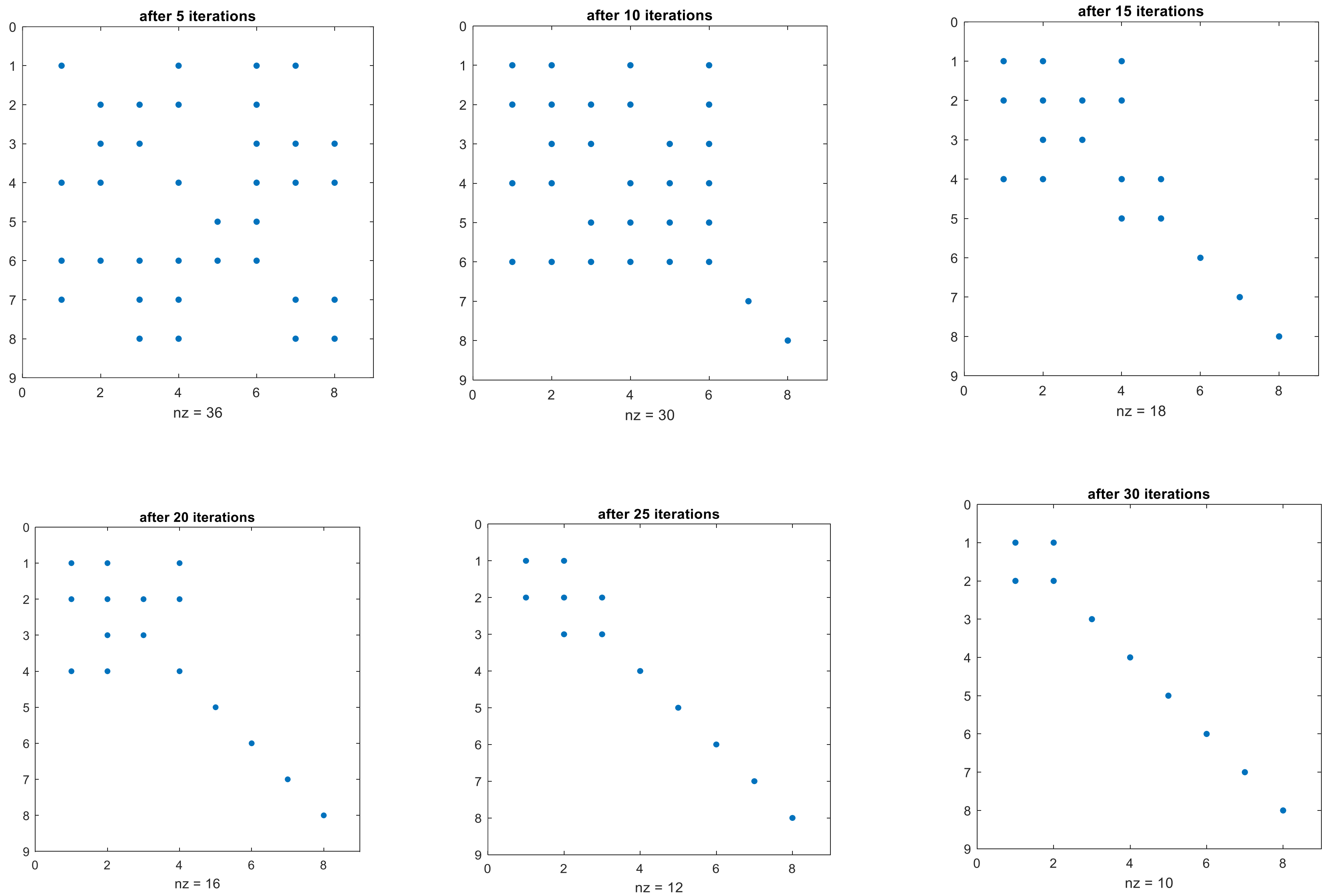
$$A_2 \equiv R_1 Q_1 \qquad A_2 = Q_2 R_2$$

$$\ldots\ldots$$

Questo alg costa o(4*(n^3/3))
4 volte n alla terza / 3 AD OGNI PASSO...
Costa parecchio quindi tutto sta nella
velocità di convergenza.

```matlab
% script QRMethod implements few steps of the basic QR
% method for computing the eigenvalues of a matrix.
% It uses, for simplicity, a random symmetric
% square matrix
A = randi([-20 20],8,8); A = A'*A;
eigenval = eig(A);
for i=1:30
    [Q R] = qr(A);
    A = R*Q;
    if mod(i,5) < 1
        A(A<1e-8) = 0;
        figure(i), spy(sparse(A))
      title(['after ',num2str(i),' iterations'])
    end
end
[eigenvalues(end:-1:1) diag(A)]
```

Questo è il codice matlab.


after 5 iterations
nz = 36
after 10 iterations
nz = 30
after 15 iterations
nz = 18
after 20 iterations
nz = 16
after 25 iterations
nz = 12
after 30 iterations
nz = 10

```
[eigenvalues(end:-1:1) diag(A)]

ans

1.0e+03 *

    2.8465    2.8465
    2.0732    2.0732
    1.2694    1.2688
    0.7981    0.7987
    0.5767    0.5767
    0.2048    0.2048
    0.0226    0.0226
    0.0038    0.0038
```

Questo teorema afferma che data una qualunque matrice non quadrata, A MxN,esistono due matrici ortogonali (quadrate a colonne ortonormali), la prima U MxM e V NxN tale che U trasposto * A*V = alla matrice diagonale dove gli elementi diagonali sono denominati con le lettere sigma

SVD Factorization

Singular Values Decomposition

Theorem: let A be any matrix, $A \in \Re^{m \times n}$, then there are two **orthogonal matrices** $U \in \Re^{m \times m}, V \in \Re^{n \times n}$ such that

$$\boxed{U^T A V = \Sigma \equiv diag\left(\sigma_1, \sigma_2, \ldots, \sigma_p\right)}$$

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0 \qquad p = \min(m, n)$$

i SIGMA sono reali e non negativi ordinati in ordine decrescente. In seguito a questa relazione di ordine che se per un certo indicie un certo indice è = a 0 allora anche i rimanenti saranno uguali a 0, cioè quelli con indice maggiore.
La matrice contenete i sigma è detto sigma maiuscolo e il teorema mi dice che A è = a U*sigma*v traposto,

$$\boxed{A = U \Sigma V^T}$$

Quindi A è scritto come prodotto di tre matirci.

Vediamo nel caso in cui sigma è MxN, diciamo che in generale la fattorizzazione esiste per ogni matrice, la cosa importante è che U e V siano ortogonali E SIGMA SIA DIAGONALE MXN QUINDI stesso ordine di grandezza di a.

SVD Factorization

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

the SVD factorization does exist for any matrix

U and V are **orthogonal**

Σ diagonal mxn

$$\begin{pmatrix} \sigma_1 & 0 & & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ & \ddots & \ddots & 0 \\ \vdots & & & \sigma_n \\ & & & 0 \\ 0 & \cdots & & 0 \end{pmatrix}$$

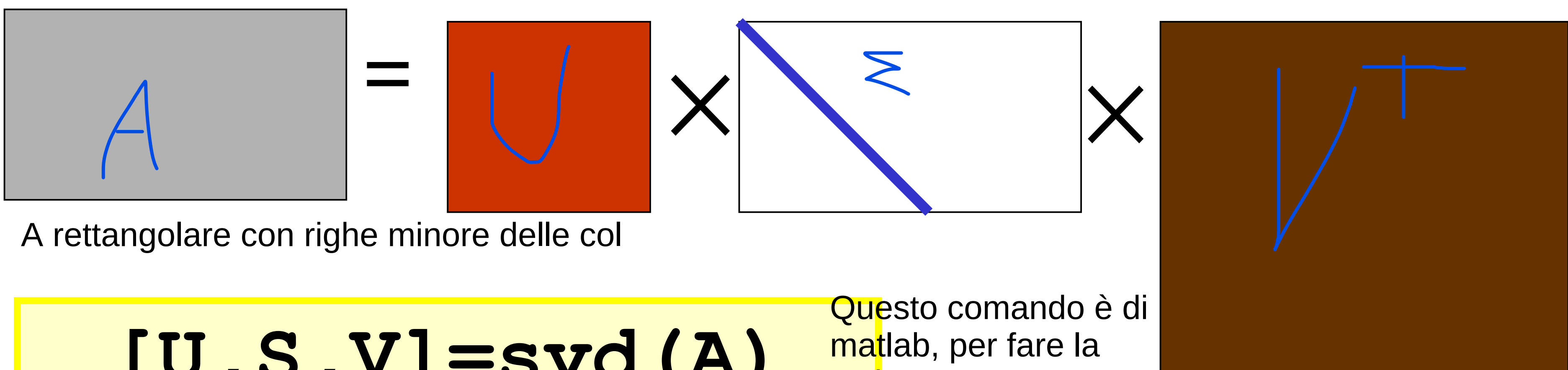
Il prof afferma che se calcolo U SIGMA E V^T so tutto di A, sa tutto nel senso che

A rettangolare

U è più grande di V

Qui ci ricorda come devono essere le dimensioni delle matrici, U è dell'ordine delle righe di A, v è delle colonne e sigma è come A, ma è retognale.

$$A\ m\times n\ ,\ U\ m\times m\ ,\ \Sigma\ m\times n\ ,\ V\ n\times n$$



A rettangolare con righe minore delle col

```
[U,S,V]=svd(A)
```

Questo comando è di matlab, per fare la svd.

un po di terminologia: i veittori colonna di U sono i vettori singolari di sinistra indicati con ui, i vettori colonna di V (cioè i vettori riga di V trapsosto) sono i vettori singolari di destra e li indichiamo con v con i. Gli elementi diagonali di sigma sono i valori singolari.

SVD Factorization

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

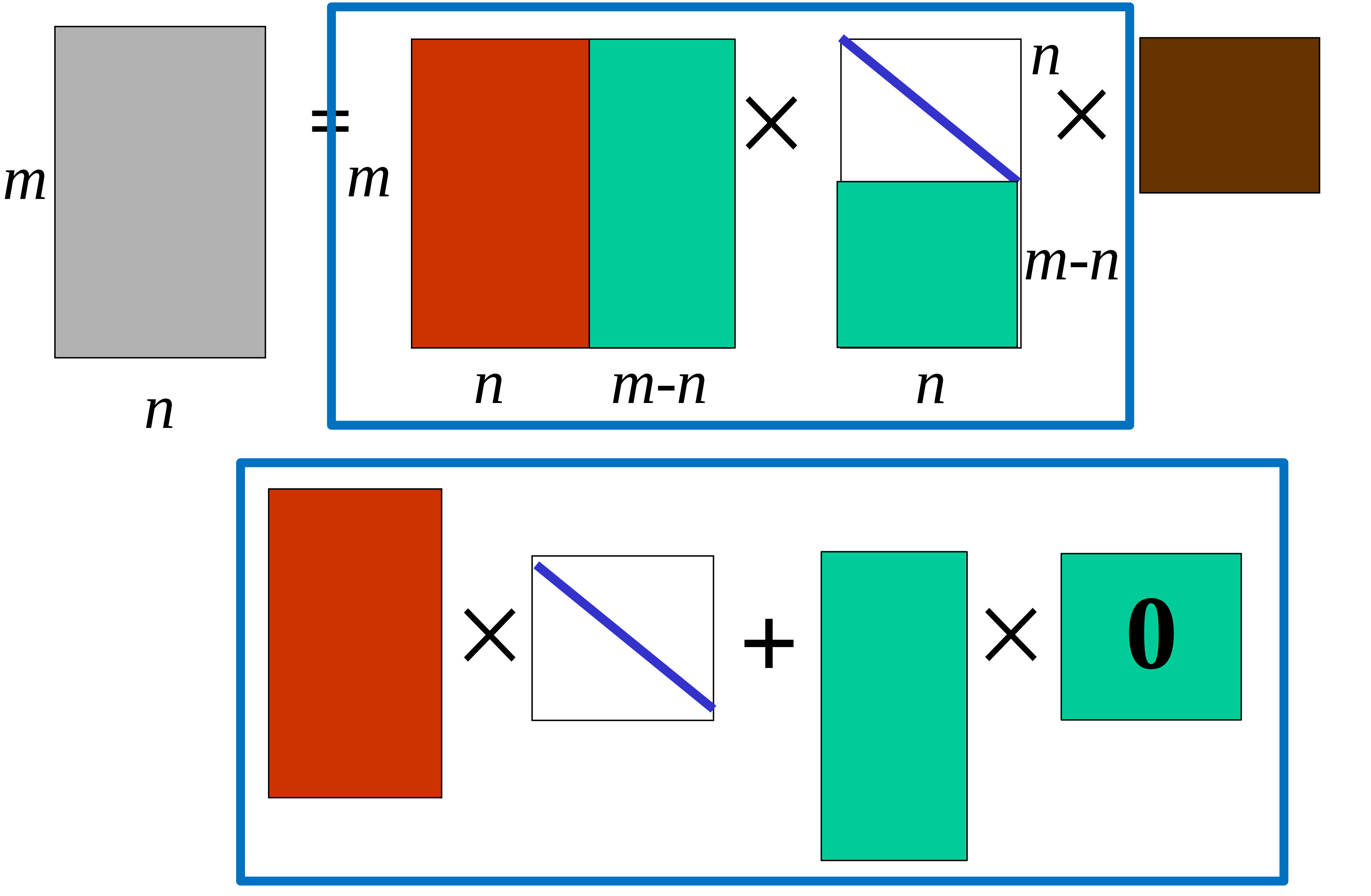
the column vectors of U are the **left singular vectors** (u_i *left singular vectors*)

the column vectors of V (i.e. row vectors of V^T) are the **right singular vectors** (v_i *right singular vectors*)

the diagonal entries of Σ are the **singular values**

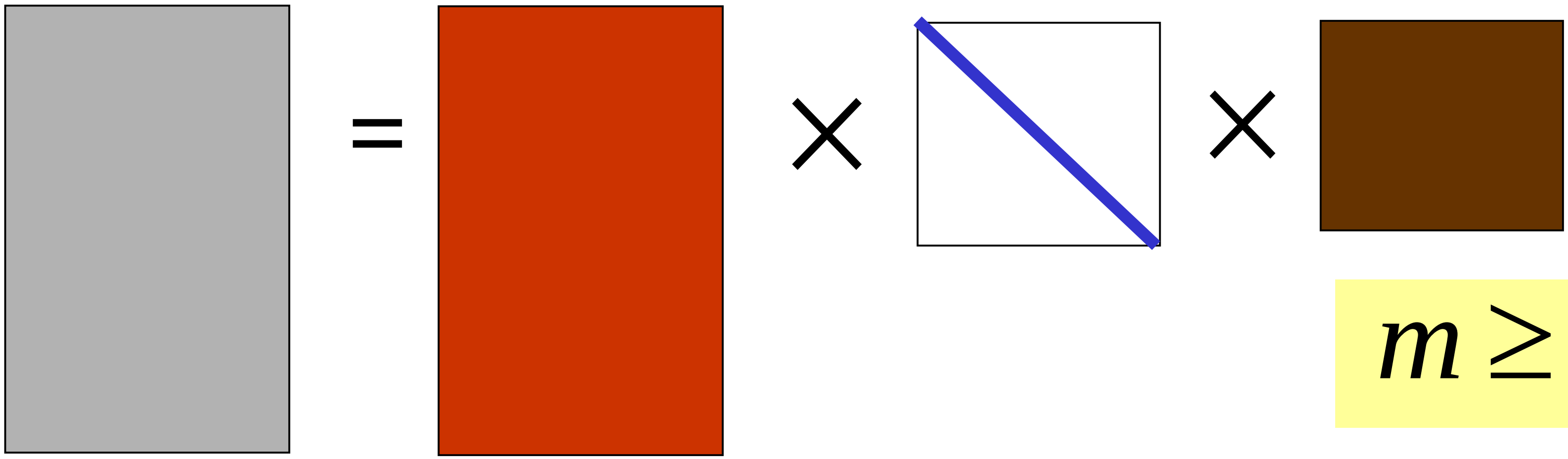
Come per la fattorizzazione QR anche qui abbiamo la versione economia proprio perché avendo in sigma m-n elementi nulli allora avremo anche m-n colonne di u quindi non le consideriamo.

SVD Factorization : reduced version (economy)



In questa slide ci mostra semplicemente che per la versione economi dobbiamo usare il comando matlab svd con lo 0 come secondo parametro che indica la versione economy.

SVD Factorization : reduced version (economy)



$$m \geq n$$

Ovviamente le matrici hanno le dimensioni diverse tenendo conto dei blocchi nulli tolti quindi appunto nella slide vi sono le nuove dimensioni.

$$A \in \Re^{m \times n}, U_n \in \Re^{m \times n}, \Sigma_n \in \Re^{n \times n}, V_n \equiv V \in \Re^{n \times n}$$

V rimane inalertata, inoltre non essendo quadrata è una matrice a colonne ortonormali (non ortogonali)

$$\boxed{A = U_n \Sigma_n V_n^T}$$

La formula è la ricostruzione di A nel caso economico (non cambia nulla).

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

Queste sono le proprietà dei valori singolari: la prob fondamentale che ha conseguenze su tutte le applicazioni della svd è che i sigma sono tutti reali e non negativi e in ordine decrescente, inoltre vale che se

SVD Factorization

properties of **singular values** :

$$\begin{pmatrix} \sigma_1 & 0 & & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ & \ddots & \ddots & 0 \\ \vdots & & & \sigma_n \\ & & & 0 \\ 0 & \cdots & & 0 \end{pmatrix}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(n,m)} \geq 0$$

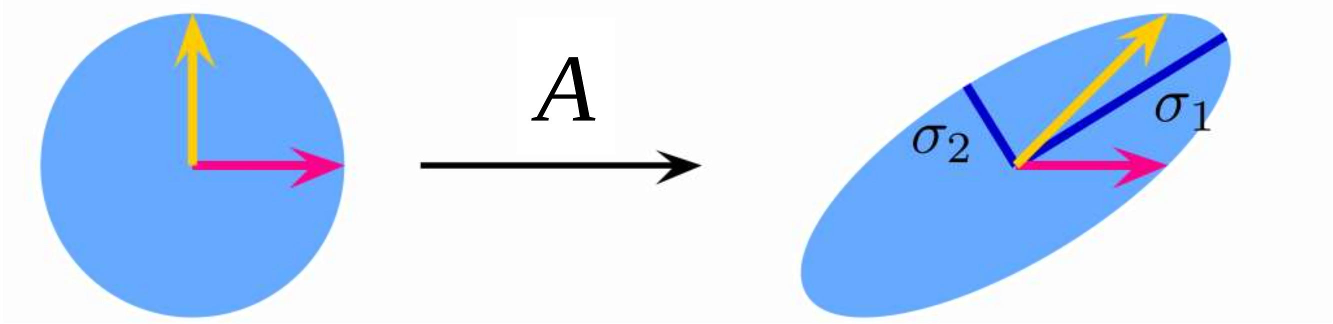
per un r sigma è diverso da 0 allora avremmo che tutti gli r-1 sono anch'essi diversi da 0 ma se per un r+1 abbiamo che è = a 0 allora tutti i successivi saranno = a 0, questa proprietà rende immediata la determinazione del rango di A, in questo caso è r perché è il numero di valori singolari strettamnte maggiore di 0.

$$\sigma_r \neq 0,\ \sigma_{r+1} = 0 \Leftrightarrow rank(A) = r$$

$$\|A\|_2 = \sigma_1 \qquad \|A\|_F = \sqrt{\sum_{i=1}^{\min(n,m)} \sigma_i^2} \qquad \kappa(A) = \frac{\sigma_1}{\sigma_n}$$

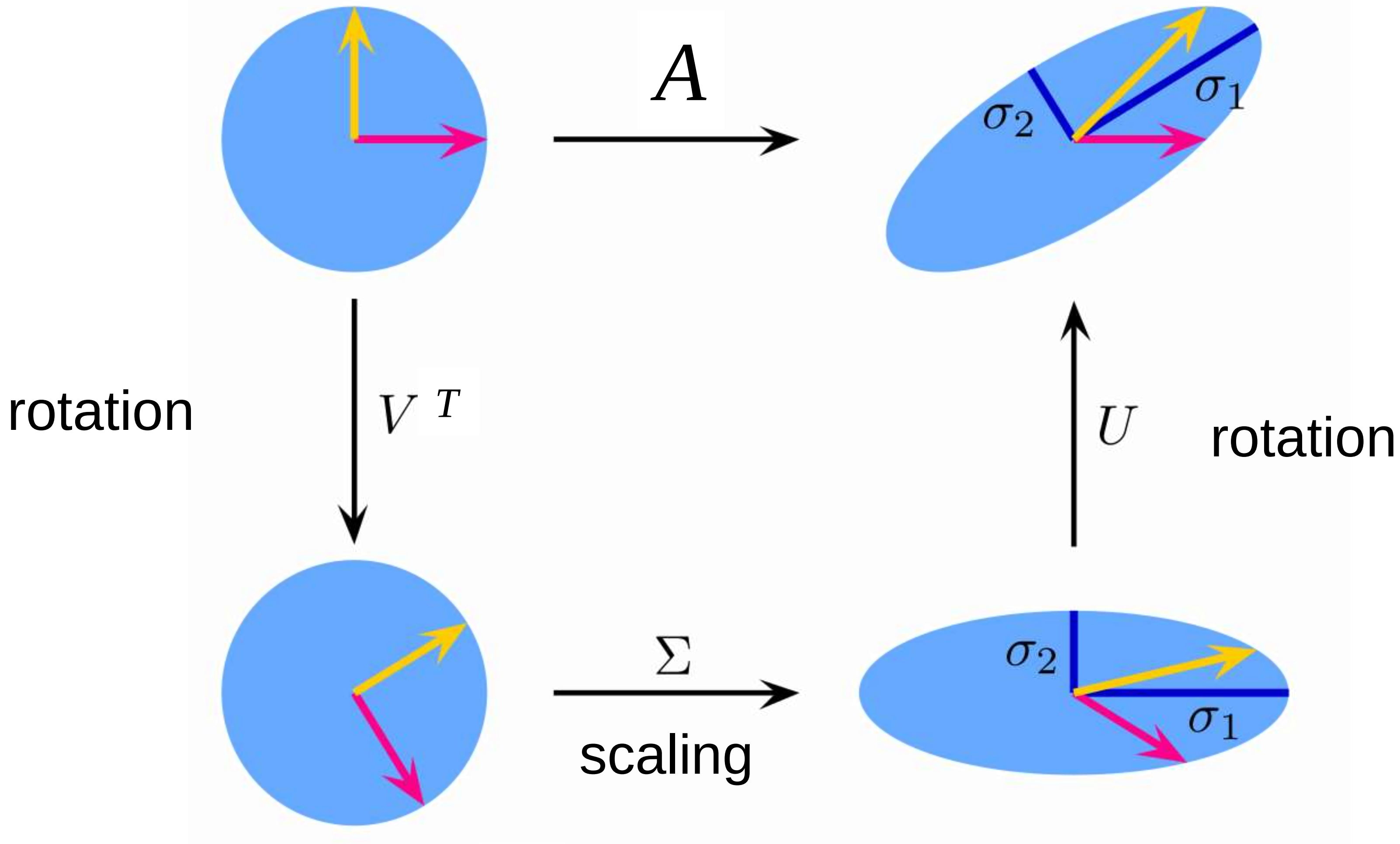
un altra proprietà è che per la norma 2 di A essa è = al primo elemento diagonale, vale inoltre che per la norma di frobenius vale che essa è la radice quadrata della somma dei quadrati di tutti i valori singolari.
L'indice di condizionamenti si A è il rapporto tra il più grande ed il più piccolo dei valori singolari.
(min di m n vuol dire prendere il più piccolo tra i due perché dipende dal caso in cui ci troviamo

the effect of A is decomposed as a sequence of an initial rotation, a scaling and a final rotation



La matrice A di fatto trasforma vettori in altri vettori, in particolare se ad A gli diamo i vettori che appartengono al suo cerchio unitario (quello blu) allora A prende i vetotri e li trasforma in altri vettori, noi possiamo vedere l'azione di A su tutto l'insieme infinito di vettori che costituiscono la sua sfera unitaria che.è la norma 2 cioè il cerchio unitario. "Se faccio il giro sugli infiniti vettori del cerchio unitario la matrice A si trasforma in un ellise".
I vettori che appartengono al cerchio unitario sono espressi(le componenti) rispetto alla base canonica di cui qui vediamo i versori (la freccia gialla e quella rossa in alto a sx). La fattorizzazione SVD consente di vedere la trasformazione provocata da A come la composizioone di varie trasformazioni, in particolare poichè U e V sono ortognali sono matrici di rotazione (o riflessione) quindi di fatto la prossima slide dice supponiamo di moltiplicare Ap quindi= USIGMA E VT * p, ma vt * p è una rotazione di p, quindi vt ruota tutti i vettori di un certo agolo (tutti i vwttori del cerchio unitario.).
i VETTORI ruotati poi sono molt per sigma che è una matrice diagonale, questo vuol dire scalare le componenti del vettore (poichè gli ele in diag sono in ordine decrescente allora le prime componenti di vp si allungano), quest'ultimo vettore ottenuto subisce un ulteriore rotazione con U. Quindi A la vediamo come tre operazioni in fila

the effect of A is decomposed as a sequence of an initial rotation, a scaling and a final rotation



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

SVD Factorization and eigenvalues

Vediamo il legame tra SVD e gli autovalori. Parto dalla classifica definita A = USIGV^t. A non è quadrata quindi non ha ne autovalore ne autovettori, ma il legame è con gli autovalori e autovettori delle due matrici quadrate A^t * A e A *A^t. Partiamo da At*A, questa matrice è nxn quindi quadrata e con la fattorizzazione SVD, quindi è la classifica formula perchè A vale sempre u sigma v traposto il tutto * la trasposta di A che è ancora una volta quella roba ma trasposta, ovviamente V non è più trasposta (Ut*U è la matrice identità).

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

$$A^T A = \left(V\Sigma^T U^T\right) U\Sigma V^T$$

Questa formula la possiamo scrivere anche come le due formule in verde sotto a quella più generica.

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

$$A^T A = V\Sigma^2 V^T$$

nxn symmetric

$$A^T A V = V\Sigma^2$$

Moltiplicando ambo i membri da destra * V allora a dx viene V^t*V che è la matr identità e la togliamo e rimane la formula nella slide in giallo, in questo modo abbiamo dimostrato il legame tra svd e gli autovalori, perché Nelle colonne di V che indichiamo con vi ci sono gli autovettori di A^T*A. N.B V è la matrice degli autovettori. Sigma quadro invece è la matrice diagonale degli autovalori di At * A

the **columns** (v_i) of V are the **eigenvectors** di A^TA

Quindi i quadrati dei valori singolari di A sono gli autovalori di A^t * A

the **squares of singular values** of A are the **eigenvalues** of A^TA

Questa è la decomposizione spettrale di B (B = A^T * A) con V matrice degli autovettori e sigma quadro matrice degli autovalori.

SVD Factorization and eigenvalues

Ora vediamo il caso di A*A^T, che è una matrice simmetrica mxm, l'idea è come quella di prima, qui però dobbiamo fare attenzione al fatto ce (VT*V è SEMPRE L'IDENTITà!) Sigma * Sigma trasposto, se sigma è mxn con m > n, sigma trasposto è nxm quindi facendo il prodotto ottengo la matrice mxm, che è più grande, ma sulla diagonale ci sono solo i primi n elementi non nulli e questo crea qualche problema, in ogni caso chiamiamo sigma^2 questa matrice ma ha una barra proprio per denotare che non è uguale a quella di prima perché questa è mxm prima era nxn e non avevamo problemi.

$$AA^T = U\Sigma V^T \left(V\Sigma^T U^T\right)$$

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

In ogni caso moltiplicando U a sx e a dx ottengo che a dx si elidono u traposto e u che è l'identità.

$$AA^T = U\Sigma\Sigma^T U^T$$

$$AA^T = U\overline{\Sigma}^2 U^T$$

mxm symmetric

$$AA^T U = U\overline{\Sigma}^2$$

Ancora una volta ottengo la composizione spettrale di B (= A*A^t) cn U matrice degli autovettori e Sigma quadro matrice degi autovalori.

the **columns** (u_i) di U are the **eigenvectors** AA^T

the **squares of singular values** of A are the **eigenvalues** of AA^T

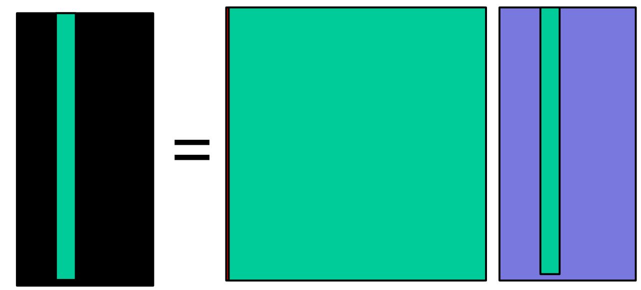
SVD Factorization

Il prof dice che è una parte un po intrigata, questi sono virtuosismi di come interpretare questi pord tra matrici.
Concentriamoci prima sul prodotto ou. V/t la posso indicare come nella slide, cioè partizionata per righe, quando la moltiplico per sigma allora la prima riga di V^t è molt per sigma 1 la seoncda riga per sigma 2 etc.
Quindi la molt da sinistra per sigma con una matr diagonale scala le righe secondo gli elementi sulla diagonale.
Ovviamente quegli elementi a zero sono gli elementi del blocco nullo di sigma.

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

bases

$$V^T = \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix} \qquad \Sigma V^T = \begin{pmatrix} \sigma_1 v_1^T \\ \sigma_2 v_2^T \\ \vdots \\ \sigma_n v_n^T \\ \boxed{\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}} \end{pmatrix}$$



Questa matrice la otteniamo per righe ma possiamo organizzarla per colonne e le chiamo s1 s2 sn. Questa matrice la moltiplichiamo secondo la formula generale, da sx per U, A= U* (s1,s2...sn) è un prod tra matrice e un vettore s e otteniamo quindi un vettore? (vedere cosa si ottiene)Us1,Us2 etc.
Possiamo concludere quindi che ogni colonna di A è fatta dalla combinazione lineare delle colonne di U, ma U sono colonne ortonormali quindi di fatto dentro alle colonne di U c'è la base per il range di A.

$$\Sigma V^T \equiv (s_1, s_2, \ldots, s_n) \quad \text{columnwise}$$

$$A = U(s_1, s_2, \ldots, s_n) = (Us_1, Us_2, \ldots, Us_n)$$

$$A \equiv (a_1, a_2, \ldots, a_n) = (Us_1, Us_2, \ldots, Us_n)$$

each column of A is a linear combination of the columns of U

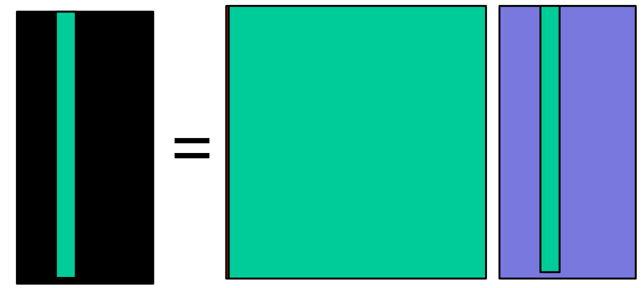
Quindi gli elementi del vettori Si sono le coordinate della i-sima colonna di A nella base costituita dalle colonne di U.
Quindi le colonne di A sono facilmente determinabili perché basta calcolare sigma * V^T e prendere le colonne.

SVD Factorization

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

bases

$$V^T = \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix} \qquad \Sigma V^T = \begin{pmatrix} \sigma_1 v_1^T \\ \sigma_2 v_2^T \\ \vdots \\ \sigma_n v_n^T \\ \boxed{\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}} \end{pmatrix}$$



Attualmente abbiamo visto che le colonne di A si ottengono combinando linearmente le colonne di U. Potremmo pensare che sia la stessa cosa della fattorizzazine QR, ma ci ricorda che nella QR non abbiamo info sulle righe. Con la prossima analisi avremo tutta l'info necessaria.

$$\Sigma V^T \equiv (s_1, s_2, \ldots, s_n)$$

columnwise

$$A \equiv (a_1, a_2, \ldots, a_n) = (Us_1, Us_2, \ldots, Us_n)$$

$$\boxed{a_i = Us_i}$$

the elements of the vector s_i are the co-ordinates of the i-th column of A on the basis formed by the columns of U

SVD Factorization

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

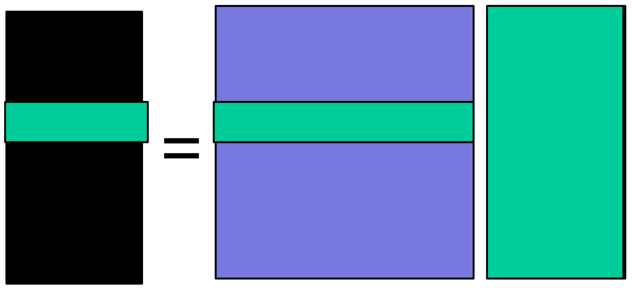
Consideriamo USigma. Parto da A partizionata per righe, poichè sigma sta a dx in questo caso scala le colonne, prima scalava le righe. Quindi la prima colonna è sigma 1 per u1, n-esima colonna sigma n é un. Similmente come prima non mi interessa che sia per colonne ma in questo caso lo prend per righe e chiamo ~~yt dve~~ yt1 è la prima colonna ma traposta quindi partizionata per riga.

bases

partitioning by rows

$$A = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_m^T \end{pmatrix}$$

$$U\Sigma = (\sigma_1 u_1, \ldots, \sigma_n u_n) = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_m^T \end{pmatrix}$$



Ricordiamo una proprietà del prod matrice * matr che dice che la prima riga di A (in generale la i-esima riga di A) è una combinazione lineare delle righe di V^T, cioè le colonne di V, usando come coefficienti gli elementi delle righe di y.

$$\alpha_i^T = y_i^T V^T$$

$$A \equiv \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_m^T \end{pmatrix} = \begin{pmatrix} y_1^T V^T \\ y_2^T V^T \\ \vdots \\ y_m^T V^T \end{pmatrix}$$

each row of A is a linear combination of the rows of V^T, i.e. of the columns of V

$$\alpha_i = V y_i$$

Per ottenere le colonne di A basta moltiplicare V * yi

SVD Factorization

Quindi possiamo concludere che gli elementi del vettore yi sono le coordinate della i-sima riga di A nellla base costituita dalle colonne di V.

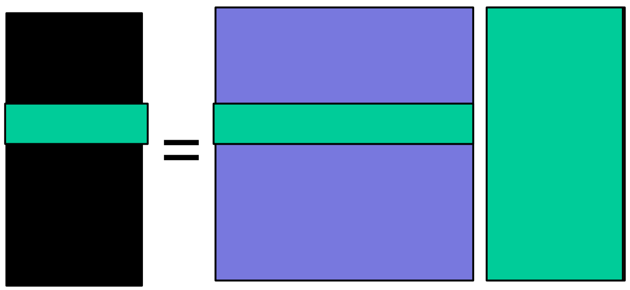
$$A = U\Sigma V^T$$

bases

partitioning by rows

$$A = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_m^T \end{pmatrix}$$

$$U\Sigma = (\sigma_1 u_1, \ldots, \sigma_n u_n) = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_m^T \end{pmatrix}$$



$$\alpha_i^T = y_i^T V^T$$

$$A \equiv \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_m^T \end{pmatrix} = \begin{pmatrix} y_1^T V^T \\ y_2^T V^T \\ \vdots \\ y_m^T V^T \end{pmatrix}$$

the elements of the vector y_i are the coordinates of the i-th row of A on the basis formed by the columns of V

$$\alpha_i = V y_i$$

Ora supponiamo che A abbia rango r: allora la mtrice U r formata dalle prime r colonne di U e la matrice Vr formata dalle prime r rine di R avremmo che vale uqella formula nella slide in vedrde do però Sigma r è formate ovviamente dai primi r elementi diagonali.

SVD Factorization

rank

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

if the rank of A is r, then the matrix U_r, formed by the first r columns of U, and the matrix V_r, formed by the first r columns of V, are such that

$$A = U_r \Sigma_r V_r^T$$

$$\Sigma_r = diag(\sigma_1, \ldots, \sigma_r)$$

Quindi possiamo ricostruire A con solo gli R elementi di queste matrici. Anche perché gli r+1 esimo in poi in Sigma annulla tutto il prodotto.

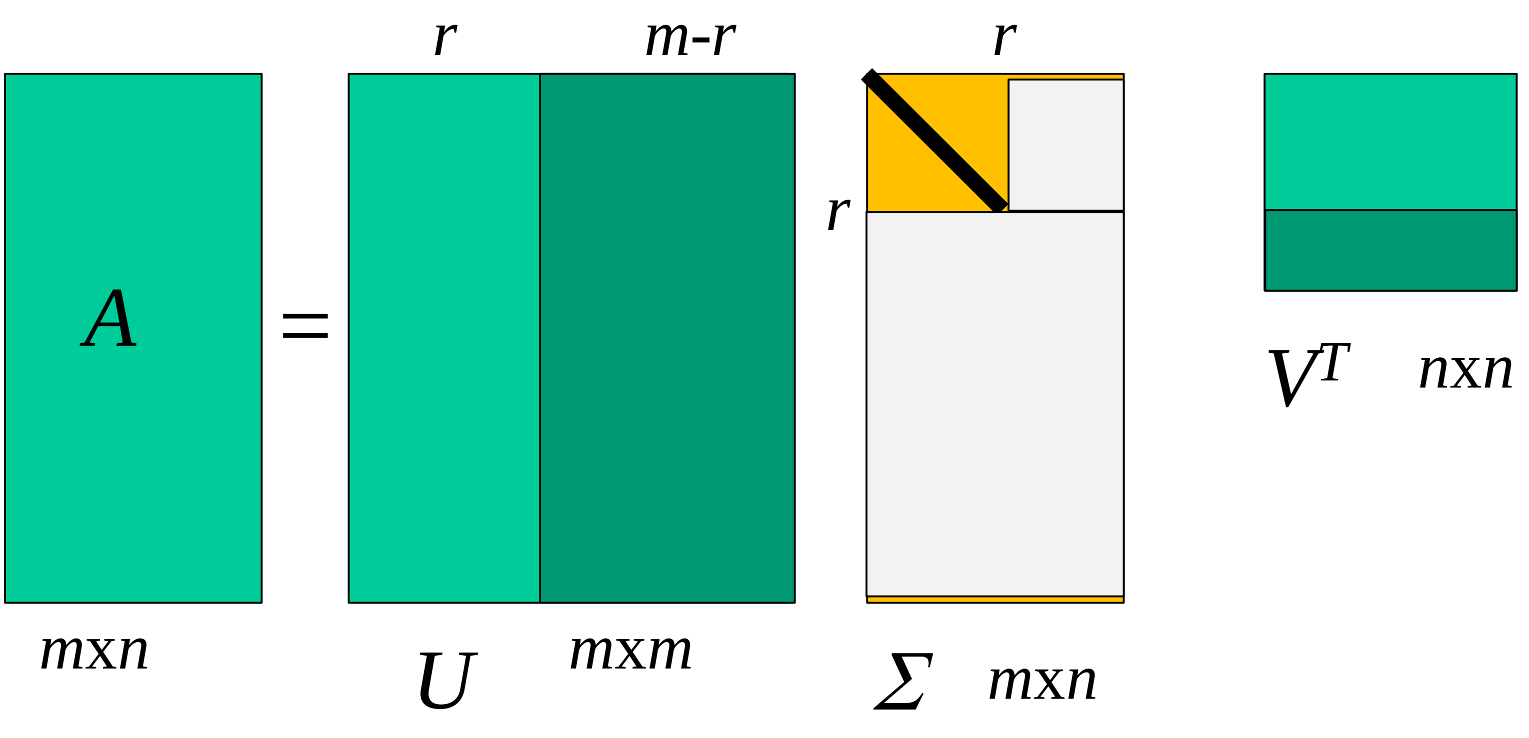
Questo è solo un esempio di quello che ha detto nella slide prima.

SVD Factorization

rank

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

$$A = U_r \Sigma_r V_r^T \qquad \Sigma_r = diag(\sigma_1, \ldots, \sigma_r)$$



Questo è solo un esempio di quello che ha detto nella slide prima. Chiaramente cambia l'indice di colonna che non è più n ma R.

SVD Factorization

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

rank

$$rank(A) = r$$

$$A = U_r \Sigma_r V_r^T$$

$$U_r = (u_1, u_2, \ldots, u_r) \in \Re^{m \times r}$$

$$\Sigma_r = diag(\sigma_1, \sigma_2, \ldots, \sigma_r,) \in \Re^{r \times r}$$

$$V_r = (v_1, v_2, \ldots, v_r) \quad \in \Re^{n \times r}$$

Questo è solo un esempio di quello che ha detto nella slide prima.

SVD Factorization

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

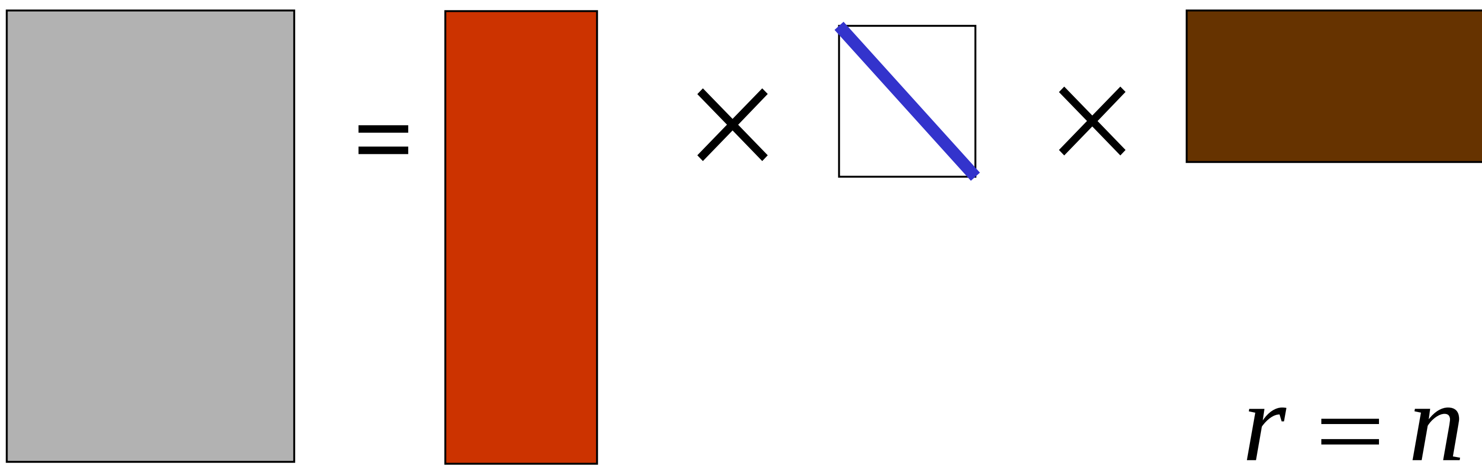
$$m \geq n$$

rank

$$rank(A) = r$$

$$A = U_r \Sigma_r V_r^T$$

example



$$r = n/2$$

Abbiamo visto la fattorizzazione economy, la standard e quella in cui prendiamo il rango r come colonne di U e r colonne di V e il blocco quadrato non nullo grande R, questa con r utile se siamo interessato solo alla base dello spazio delle righe e delle oclonne, vediamo ora che U e V hanno anche altre info. Come detto le prime r colonne di U sono una base ortornomale dello spazio delle colonne di A cioè di range(A). Le prime r colonne di V osno una base ortornomale dello spazio delle righe di A cioè di range(A^t). Quindi ho immediatamente i projettori ortogonali sullo spazio delle colonne che è Ur*U^t mentre per le righe è Vr*V^t

SVD Factorization

bases

$A = U\Sigma V^T$

the first r columns of U (matrix U_r) are an **orthonormal basis of the column space** of A, i.e. of the $range(A)$

the first r columns of V (matrix V_r) are an **orthonormal basis of the row space** of A, i.e. of the $range(A^T)$

an **orthogonal projector** onto the $range(A)$ is

$$U_r U_r^T$$

an **orthogonal projector** onto the $range(A^T)$ is

$$V_r V_r^T$$

Qui non aggiunge niente di che, alla fine dice che otteniamo info come nella fattorizzazzione QR ma di fatto qui abbiamo anche info sulle righe.

SVD Factorization

bases

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

the first r columns of U (matrix U_r) are an **orthonormal basis of column space** of A, i.e. of the $range(A)$

$$A = U_r \Sigma_r V_r^T = U_r (\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n)$$

$$\boxed{A \equiv (a_1, a_2, \ldots, a_n) = (U_r \bar{s}_1, U_r \bar{s}_2, \ldots, U_r \bar{s}_n)}$$

$$\boxed{a_i = U_r \bar{s}_i}$$

$\bar{s}_i$ is the vector formed by the first r components of s_i

Come visto prima le righe di A lo otteniamo con la formula verde in basso.

Qui non aggiunge nulla.

SVD Factorization

bases

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

the first r columns of V (matrix V_r) are an **orthonormal basis of the row space** of A, i.e. of the $range(A^T)$

$$U_r\Sigma_r = (\sigma_1 u_1, \dots, \sigma_r u_r) = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_m^T \end{pmatrix}$$

$$A \equiv \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_m^T \end{pmatrix} = \begin{pmatrix} y_1^T V_r^T \\ y_2^T V_r^T \\ \vdots \\ y_m^T V_r^T \end{pmatrix}$$

$$\boxed{\alpha_i^T = y_i^T V_r^T}$$

$$\boxed{\alpha_i = V_r\, y_i}$$

Ritornando alla fattorizzazione "completa", abbiamo visto prima che le prime r colonne di U e V sono una base ortogonale per lo spazio delle righe e delle colonne. Ma le ultime m-r colonne di U e V?

SVD Factorization

bases

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

le ultime m-r colonne di U sono una base ortonormale del complemento ortogonale dello spazio delle colonne di A cioè dello spazio nullo di A trasposto.

the last m-r columns of U are an **orthonormal basis of the orthogonal complement of the column space** of A, i.e. of the null space of A^T

le ultime n-r colonne di V sono una base ortonormale del complemento ortogonale dello spaziodelle righe di A, cioè lo spazio nullo di A

the last n-r columns of V are an **orthonormal basis of the orthogonal complement of the row space** of A, i.e. of the null space of A

Spesso possiamo trovare altri modi di identificare la decomposizione spettrale. Moltiplichiamo V ambo i membri e otteniamo 2. A questo punto usiamo il prod matrice per matrice decomponendo con le colonne di vi il tutto è = alla i-esima col di u * sigma i otteniamo quindi 3. Questo definizione ha un analogia con la definizione di autovalori e autovettori. . LO STESSO VALE PER LA TRASPOSTA

SVD Factorization

matrix-vector identities

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

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

2 $$AV = U\Sigma$$

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

3 $$\boxed{Av_i = \sigma_i u_i}$$

C $$\boxed{A^T u_i = \sigma_i v_i}$$

note the *analogy* with the definition of eigenvalues / eigenvectors

La slide dice praticamente che potremmo trovarci la fattorizzazione SVD in questo modo qui.

SVD Factorization

Questa è la parte centrale. Ricordiamo che se abbiamo il prod M = P*T allora M lo possiamo scrivere sommando matrici di rango 1 che si ottengono come prodotto esterno con i-esima colonna di P*iesima colonna di T,otteniamo la matrice di rango 1. Ora qui scaliamo le colonne di U moltiplicando ogni col di U per il corrispondente valore singolare. Quindi posso interpretare questo prodotto (usgima*v trasposto) come somma di matrici di rango 1 e chiamo Ei l'iesimo addendo di questa somma, cioè Ei è il valore del prodotto esterno tra sigma e U e v trasposto ovviamente Ei è solo l'i-simo elemento sia di sigma che u che v traposto. Alla fine sommiamo le matrici e otteniamo A, avremmo i matrice tante quanto è il rango di A.
E ha il rango 1.

Questa slide di fatto mostra la forza della SVD. Lui ci dice che Ei incorpora anche i sigma ma i sigma hanno un certo ordine che è decrescente, questo significa che le prime matrici hanno più importanza delle ultime, di fatto la forza è che ordinale matrici di rango 1 in senso decrescente di importanza.

sum of rank 1 matrices

$$A = \boxed{U\Sigma} V^T$$

interpretation of the product between the matrix $U\Sigma$ and the matrix V as the sum of external products (column x row)

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

rank 1 matrix

$$A = \sum_{i=1}^{r} E_i$$

Su questa slide non si è soffermato perchè dice quello detto nella slide di prima.

SVD Factorization

sum of rank 1 matrices

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

$$A = (\sigma_1 u_1, \sigma_2 u_2, \ldots, \sigma_n u_n) V^T$$

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \ldots + \sigma_n u_n v_n^T$$

if **rank** = r

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

$$A = \sum_{i=1}^{r} E_i$$

any matrix of rank r can be written as the **sum** of r **rank 1 matrices**

SVD Factorization

Ovviamente avremmo potuto fare la stessa cosa anche se A è trasposto ponendo Ei traposto = Fi

sum of rank 1 matrices

$$A^T = \boxed{V\Sigma^T} U^T$$

interpretation of the product between the matrix $V\Sigma^T$ and the matrix U^T as sum external products (column x row)

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

rank 1 matrix

$$A^T = \sum_{i=1}^{r} F_i$$

$$F_i = E_i^T$$

Abbiamo visto prima che con l'interpretazione del prodtto esterno possiamo vedere come somma di R matrici di rango 1, con r costruisco esattmanete R. Ma se ne prendo un k numero più piccolo di R? Allora avrà che Ak è un approssimazione di A, Ak è una matrice di rango K.
Ak la possiamo scirvere nei due modi della slide.

k-truncated SVD Factorization

$$A^{(k)} = \sum_{i=1}^{k} E_i$$

$$k < r$$

$A^{(k)}$ is an approximation of A

$A^{(k)}$ is a matrix of rank k

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

$$U_k = \left(u_1, u_2, \ldots, u_k\right)$$

$$\Sigma_k = diag\left(\sigma_1, \ldots, \sigma_k\right)$$

$$V_k = \left(v_1, v_2, \ldots, v_k\right)$$

LA AK è la migliroe approssimazione di rango k di A. Inoltre nella slide calcola anche la differenza tra le due A, in norma di frobenius è la formula con la radice quadrata, cioè somma dei quadrati di sigma che vanno da k+1 fino ad r cioè sono i valori che abbiamo trascurato , mentre in nomra due è sigma k+1. Quindi questo è l'approssimazione dell'errore . In nomra due la differenza è esattmanete il primo valore trascurato.

k-truncated SVD Factorization

$k < r$

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

$$A^{(k)} = \sum_{i=1}^{k} E_i$$

$A^{(k)}$ is the **best approximation** of rank k of A

$$\left\| A - A^{(k)} \right\|_2 = \sigma_{k+1}$$

approximation error

$$\left\| A - A^{(k)} \right\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}$$

Qui da giusto degli accenni, in particolare ci dice che il calcolo delle matrice è ben condizionato

SVD Factorization

algorithms

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

calculation of the matrices U, Σ, V

well conditioned problem

Ed esistono algirtmi abbastanza sofisticati, il primo è il caclolo degli autovalori e degli autovettori di A^T * A, ma la cosa più efficiente è Golub Reinsch la cui complessità è quella nella slide. Quest'ultimo algoritmo è usato da svd di matlab.

SVD Factorization

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

algorithms

through the calculation of the eigenvalues and eigenvectors of A^TA

or, more efficient:

Golub-Reinsch algorithm

time complexity $O(mn^2+n^3)$

cubicaa

SVD Factorization

Qui mostra il calcolo degli autovalori e autovettori ma dice che non vuole rivederlo perché l'ha gia fatto all'inizio nella lezione apposita.

algorithms

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

eigenvalues / eigenvectors of A^TA

$$A^TA = V\Sigma^TU^TU\Sigma V^T$$

$$A^TA = V\Sigma^2V^T$$

$$A^TAV = V\Sigma^2$$

V and Σ are computed

$$AV = U\Sigma$$

U_r is computed

Questo lo ignora.

Exercise:
develop a Matlab function that computes the SVD through the computation of the eigenvalues /eigenvectors of $A^T A$

$$A^T A V = V \Sigma^2$$

columns of V are eigenvectors

the diagonal of Σ contains the square root of the eigenvalues

$$AV = U\Sigma$$

compute AV (is a matrix) and then divide each column by the corresponding singular value (you will get U_r)

IGNORA.

Exercise:
develop a Matlab function that computes the SVD through the computation of the eigenvalues /eigenvectors of $A^T A$

```
function [U S V] = MySVDeig(A)
% compute the SVD of the matrix A
% by means of the spectral factorization of
% the square symmetric matrix A'*A
%
[X L] = eig(A'*A);
V = fliplr(X);
S = real(diag(sort(sqrt(diag(L)),'descend')));
U = (A*V)./(ones(size(A,1),1)*diag(S)');
end
```

SVD Factorization

L'algoritmo di Golub si basa sulla trasformazione ortogonale della matrice A (una successione di trasformazioni che lavorano sia su righe che colonne di A) otteniamo un matrice bidiagonale superiore che chiamiamo B.Dopodichè con un metodo iterativo trasforma B in una matrice diagonale con matrici ortogonali che ci permette di non alterare i valori singolari.

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

algorithms

the algorithm is based on the orthogonal transformation of A into an upper bidiagonal matrix B

$$U_B^T A V_B = B$$

followed by an iterative process (with orthogonal matrices) which transforms B into a diagonal matrix

$$U_\Sigma^T B V_\Sigma = \Sigma = diag\left(\sigma_1, \ldots, \sigma_n\right)$$

SVD Factorization

La composizione delle matrice U traposto sigma, V sigma, U traposto B e Vb danno luogo alle matrici U e V.ù
Questa è solo un idea di massima e non ha dato nessuna ulteriore specifica.

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

algorithms

the algorithm is based on the orthogonal transformation of A into an upper bidiagonal matrix B

followed by an iterative process (with orthogonal matrices) which transforms B into a diagonal matrix

$$U_B^T A V_B = B \qquad U_\Sigma^T B V_\Sigma = \Sigma$$

$$U_\Sigma^T U_B^T A V_B V_\Sigma = \Sigma$$

$$U = U_B U_\Sigma \quad , \quad V = V_B V_\Sigma$$

$$\boxed{U^T A V = \Sigma}$$

Questa slide mostra i software per il calcolo della SVD per le matrici piene e sparse.
Anche matlab poii mette a disposizione un comando per le matrici sparse usando l'approssimazione K.

Software for the SVD factorization

full matrices

Lapack, ScaLapack

http://www.netlib.org/lapack

http://www.netlib.org/scalapack

sparse matrices

SVDpack, SVDpackC

http://www.netlib.org/svdpack

SVD for sparse matrices in Matlab

```
[Uk,Sk,Vk] = svds(A,k)
```