SIS Scuola Interdipartimentale delle **Scienze**, dell'**Ingegneria** e della **Salute**

DiST Dipartimento di Scienze e Tecnologie

Laurea Magistrale in STN

Applicazioni di Calcolo Scientifico e Laboratorio di ACS

(12 cfu)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4
stanza: n. 423 – Lato Nord, 4° piano
tel.: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

ACS parte 2: ACS_09a

Argomenti trattati

- **Cenni di Statistica inferenziale.**

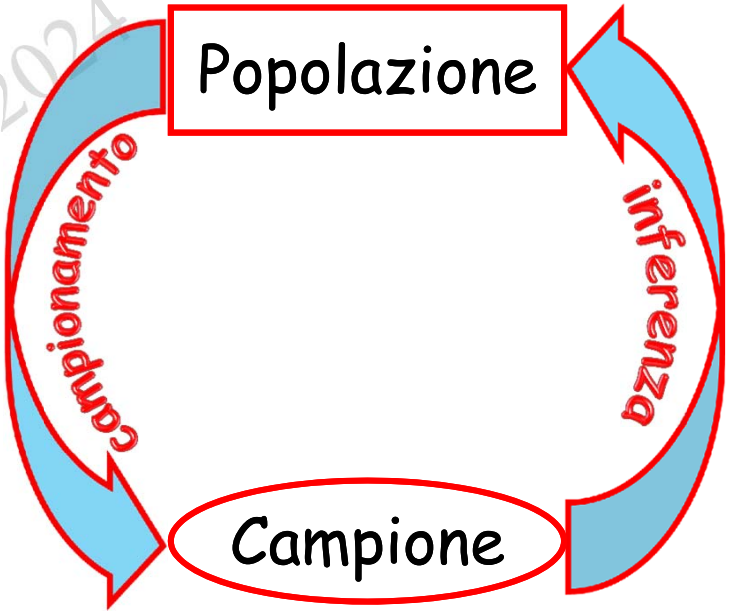
Cenni di Inferenza Statistica

La Statistica, che è una scienza quantitativa, è lo strumento idoneo per prendere decisioni in situazioni di incertezza.

La Statistica Inferenziale è l'insieme di metodi statistici che, partendo dall'osservazione e dall'analisi del "particolare" (campione), permettono di risalire al "contesto generale" (popolazione).

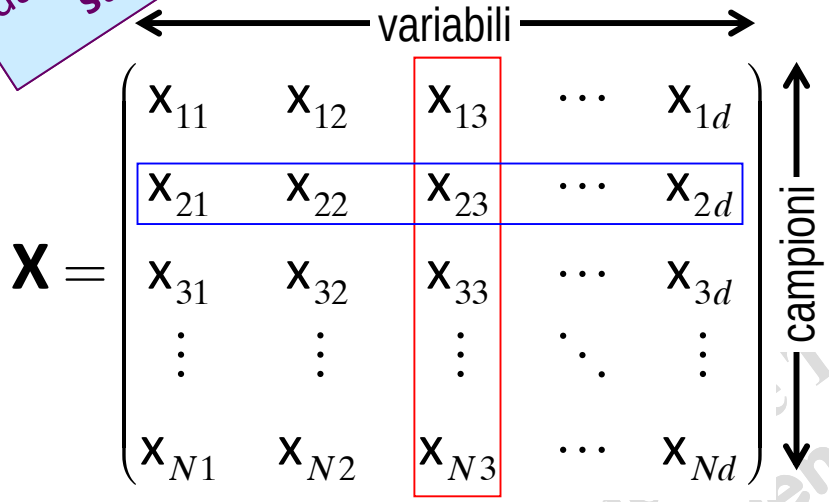
L'Inferenza Statistica si svolge secondo le seguenti fasi:

- Estrazione di una parte della popolazione (campionamento).
- Calcolo sui dati campionari di quantità (stime dei parametri) quali, ad es., media campionaria, varianza o altre.
- Estensione alla popolazione dei risultati forniti dal campione (inferenza).



Matrice dei dati X

dal punto di vista **statistico**



X è una matrice di size $N \times d$, dove:

- N è il numero dei *campioni*.
- d è il numero delle *variabili casuali* (*feature*), ed anche la dimensione dello spazio dati.

Ogni riga di **X** contiene un *campione* (*datum*).

Ogni colonna di **X** contiene una *variabile casuale* (*feature*).

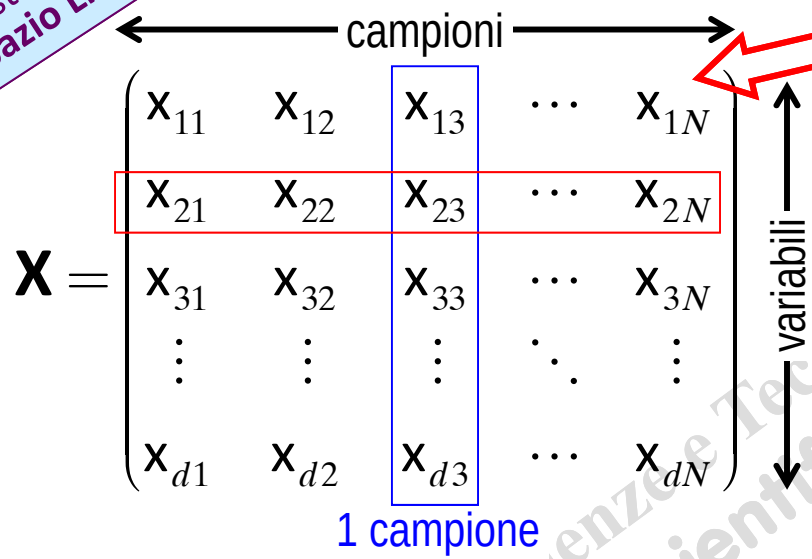
x_{ij} è la j^{sima} *variabile* scelta dall'i^{simo} *campione*.

```
X=[ -1     1     2     2
    -2     3     1     0
     4     0     3    -1 ];
N=size(X,1); % numero di campioni
d=size(X,2); % numero di variabili casuali
```

I Software Statistici si aspettano la matrice dei dati in tale formato.

Matrice dei dati X

dal punto di vista di uno **Spazio Lineare**



$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1N} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2N} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \cdots & x_{dN} \end{pmatrix}$$

trasposta della precedente matrice

X è una matrice di size $d \times N$, dove:

- N è il numero dei *campioni*.
- d è il numero delle *variabili casuali* (*v.c.*).

```
X=[-1    1    2    2
   -2    3    1    0
    4    0    3   -1]';
```

Ogni colonna di **X** contiene un *campione* (*datum*).

Ogni riga di **X** contiene una *variabile casuale* (*feature*).

x_{ij} è la i^{sima} *variabile* scelta dal j^{simo} *campione*.

Questa notazione è più vicina all'Algebra delle Matrici, perché i *campioni* sono considerati come vettori colonna, e le loro componenti sono i valori delle *v.c.*. Ciò implica che gli N campioni appartengono ad uno spazio a d dimensioni (*spazio delle variabili*).

I Software Numerici si aspettano la matrice dei dati in tale formato.

dal punto di vista di uno **Spazio Lineare**

colonne: campioni

$$\mathbf{X} = \left(\mathbf{X}^{(1)} \quad \mathbf{X}^{(2)} \quad \cdots \quad \mathbf{X}^{(N)}\right) = \begin{pmatrix} X_1^{(1)} & X_1^{(2)} & \cdots & X_1^{(N)} \\ \vdots & \vdots & & \vdots \\ X_d^{(1)} & X_d^{(2)} & \cdots & X_d^{(N)} \end{pmatrix}$$

righe: variabili

Il vettore **media campionaria** $\boldsymbol{\mu}$, ha size (d×1), ed è il vettore colonna le cui componenti sono calcolate come media dei valori di ogni riga (cioè il valor medio di ciascuna *variabile* su tutti i campioni):

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_d \end{pmatrix} : \quad \mu_i = \frac{1}{N}\sum_{j=1}^{N} X_j^{(i)} = \frac{1}{N}\sum_{j=1}^{N} x_{ij}$$

La **matrice centrata dei dati** $\mathbf{X}_C$ è ottenuta sottraendo il *vettore media campionaria* $\boldsymbol{\mu}$ da ogni campione, cioè da ogni colonna di $\mathbf{X}$:

$$\mathbf{X}_C = \begin{pmatrix} x_{11}-\mu_1 & x_{12}-\mu_1 & x_{13}-\mu_1 & \cdots & x_{1N}-\mu_1 \\ x_{21}-\mu_2 & x_{22}-\mu_2 & x_{23}-\mu_2 & \cdots & x_{2N}-\mu_2 \\ x_{31}-\mu_3 & x_{32}-\mu_3 & x_{33}-\mu_3 & \cdots & x_{3N}-\mu_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1}-\mu_d & x_{d2}-\mu_d & x_{d3}-\mu_d & \cdots & x_{dN}-\mu_d \end{pmatrix}$$

$\mathbf{X}_C = \mathbf{X} - \boldsymbol{\mu}\,\mathbf{1}_N^{\mathsf{T}}$ (prodotto esterno)

dove

$\mathbf{1}_N = (1, 1, \ldots, 1)^{\mathsf{T}}$

$\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_d)^{\mathsf{T}}$

dal punto di vista di uno **Spazio Lineare**

colonne: campioni

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(N)} \end{pmatrix} = \begin{bmatrix} X_1^{(1)} \\ \vdots \\ X_d^{(1)} \end{bmatrix} \begin{bmatrix} X_1^{(2)} \\ \vdots \\ X_d^{(2)} \end{bmatrix} \begin{bmatrix} \cdots \\ \cdots \\ \cdots \end{bmatrix} \begin{bmatrix} X_1^{(N)} \\ \vdots \\ X_d^{(N)} \end{bmatrix}$$

righe: variabili

Tramite la matrice centrata dei dati $\mathbf{X}_C$ si può calcolare:

La **Scatter matrix S** ($d \times d$): somma dei prodotti esterni dei campioni centrati $(\mathbf{x}^{(k)} - \boldsymbol{\mu})$:

$$\mathbf{S} = \mathbf{X}_C \, \mathbf{X}_C^{\mathsf{T}} = \sum_{k=1}^{N} \left(\mathbf{x}^{(k)} - \boldsymbol{\mu}\right)\left(\mathbf{x}^{(k)} - \boldsymbol{\mu}\right)^{\mathsf{T}}$$

La **matrice di Covarianza C** ($d \times d$):

$$\mathbf{C} = \frac{1}{N-1} \mathbf{X}_C \, \mathbf{X}_C^{\mathsf{T}} = \frac{1}{N-1} \mathbf{S} \quad \text{o} \quad \mathbf{C} = \frac{1}{N} \mathbf{X}_C \, \mathbf{X}_C^{\mathsf{T}} = \frac{1}{N} \mathbf{S}$$

La **matrice di Correlazione R** ($d \times d$):

$$\mathbf{R} = \frac{1}{N-1} \mathbf{X}_S \, \mathbf{X}_S^{\mathsf{T}} \quad \text{o} \quad \mathbf{R} = \frac{1}{N} \mathbf{X}_S \, \mathbf{X}_S^{\mathsf{T}}$$

dove $\mathbf{X}_S = \mathbf{D}^{-1} \mathbf{X}_C$ è la *matrice standardizzata*

e $\mathbf{D} = \mathrm{diag}\left(\sigma_1, \sigma_2, \ldots, \sigma_d\right)$ σ_j : dev. std. della j$^{\text{sima}}$ *v.c.*

$$\mathbf{x}_S = \begin{pmatrix} \frac{x_{11}-\mu_1}{\sigma_1} & \frac{x_{12}-\mu_1}{\sigma_1} & \cdots & \frac{x_{1N}-\mu_1}{\sigma_1} \\ \frac{x_{21}-\mu_2}{\sigma_2} & \frac{x_{22}-\mu_2}{\sigma_2} & \cdots & \frac{x_{2N}-\mu_2}{\sigma_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_{d1}-\mu_d}{\sigma_d} & \frac{x_{d2}-\mu_d}{\sigma_d} & \cdots & \frac{x_{dN}-\mu_d}{\sigma_d} \end{pmatrix}$$

dal punto di vista di uno **Spazio Lineare**

Esempio MATLAB

$\mathbf{X}$ (campioni X_1 X_2 X_3; variabili x_1 x_2 x_3 x_4)

$$\begin{bmatrix} -1 & -2 & 4 \\ 1 & 3 & 0 \\ 2 & 1 & 3 \\ 2 & 0 & -1 \end{bmatrix}$$

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);      % numero dei campioni
mu  = mean(X,2);    % vettore colonna delle medie
sig = std(X,0,2);   % vettore delle deviazioni std.
```

```
std(X,0)  deviazione standard normalizzata per 1/(N-1)   default
std(X,1)  deviazione standard normalizzata per 1/N
```

matrice centrata $\mathbf{X_C}$

```
Xc=X - repmat(mu,1,N);
mean(Xc,2)
ans =
  -1.4803e-16
   7.4015e-17
            0
   7.4015e-17
```

media zero

```
[sig std(Xc,0,2)]
ans =
    3.2146    3.2146
    1.5275    1.5275
         1         1
    1.5275    1.5275
```

matrice standardizzata $\mathbf{X_S}$

```
Xs=normalize(X,2);
% =zscore(X,0,2);
mean(Xs,2)
ans =
     0
     0
     0
     0
```

media zero

```
std(Xs,0,2)
ans =
     1
     1
     1
     1
```

dev. std. 1

$$\mathbf{X_S} = \mathbf{D}^{-1}\,\mathbf{X_C}$$

```
D=diag(sig);
inv(D)*Xc
ans =
   -0.41478   -0.72587    1.1406
   -0.21822    1.0911    -0.87287
          0         -1          1
     1.0911   -0.21822   -0.87287
Xs=normalize(X,2)
Xs =
   -0.41478   -0.72587    1.1406
   -0.21822    1.0911    -0.87287
          0         -1          1
     1.0911   -0.21822   -0.87287
D1=diag(1./sig);
D1*Xc  % senza inversa
ans =
   -0.41478   -0.72587    1.1406
   -0.21822    1.0911    -0.87287
          0         -1          1
     1.0911   -0.21822   -0.87287
```

dal punto di vista di uno **Spazio Lineare**

Esempio MATLAB

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % numero di campioni
mu = mean(X,2); % vettore colonna delle medie
```

$$\mathbf{X} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \text{variabili} \begin{array}{c} \text{campioni} \\ \begin{array}{ccc} X_1 & X_2 & X_3 \end{array} \\ \begin{bmatrix} -1 & -2 & 4 \\ 1 & 3 & 0 \\ 2 & 1 & 3 \\ 2 & 0 & -1 \end{bmatrix} \end{array}$$

Le funzioni MATLAB **cov()**, **corrcoef()**, **pca()**, … vogliono, come parametro, la matrice dei dati **X** in formato statistico, cioè di size: $N \times d$ (campioni × variabili)
e **non** $d \times N$ (variabili × campioni)

È necessario passare la trasposta di **X**.

```
C=cov(X')   % matrice di covarianza
C =
   10.3333   -4.1667    3.0000   -3.1667
   -4.1667    2.3333   -1.5000    0.3333
    3.0000   -1.5000    1.0000   -0.5000
   -3.1667    0.3333   -0.5000    2.3333
```

matrice centrata $\mathbf{X}_C$

```
Xc = X - repmat(mu,1,N);
Xc*Xc'/(N-1) % scatter matrix scalata
ans =
   10.3333   -4.1667    3.0000   -3.1667
   -4.1667    2.3333   -1.5000    0.3333
    3.0000   -1.5000    1.0000   -0.5000
   -3.1667    0.3333   -0.5000    2.3333
```

```
R=corrcoef(X')  % matrice di correlazione
R =
    1.0000   -0.8486    0.9333   -0.6449
   -0.8486    1.0000   -0.9820    0.1429
    0.9333   -0.9820    1.0000   -0.3273
   -0.6449    0.1429   -0.3273    1.0000
```

matrice standardizzata $\mathbf{X}_S$

```
Xs = normalize(X,2); % matrice standardiz.
Xs*Xs'/(N-1)
ans =
    1.0000   -0.8486    0.9333   -0.6449
   -0.8486    1.0000   -0.9820    0.1429
    0.9333   -0.9820    1.0000   -0.3273
   -0.6449    0.1429   -0.3273    1.0000
```

dal punto di vista di uno **Spazio Lineare**

Esempio MATLAB: PCA*

* nello Statistics and Machine Learning Toolbox di MATLAB

$\mathbf{X}$ (campioni: X_1 X_2 X_3; variabili: x_1, x_2, x_3, x_4)

$$\begin{bmatrix} -1 & -2 & 4 \\ 1 & 3 & 0 \\ 2 & 1 & 3 \\ 2 & 0 & -1 \end{bmatrix}$$

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % numero dei campioni
mu = mean(X,2); % vettore colonna delle medie
```

← Tutti i parametri di output →

```
[base,comp,lambda,tsquared,explained,m] = pca( X' ,'Economy',false);
```

- scores (λ) → `comp`
- statistica T-quadro di Hotelling → `tsquared`
- percentuale della varianza totale → `explained`

```
[explained    lambda/sum(lambda)*100]
ans =
       86.622          86.622
       13.378          13.378
            0               0
            0               0
```

percentuali

ricostruzione dei dati

```
Xc = X-repmat(mu,1,N)
Xc =
    -1.3333      -2.3333       3.6667
   -0.33333       1.6667      -1.3333
          0           -1            1
     1.6667      -0.33333      -1.3333
```

```
base*comp'
ans =
    -1.3333      -2.3333       3.6667
   -0.33333       1.6667      -1.3333
 -1.1102e-16           -1            1
     1.6667      -0.33333      -1.3333
```

proiezione dei dati

```
base'*Xc % =base\Xc
ans =
     -1.4641      -2.7682       4.2323
      1.5884      -1.2925     -0.29588
   2.2204e-16   4.4409e-16  -4.4409e-16
   2.7756e-17   1.3878e-16            0
```

```
comp'
ans =
     -1.4641      -2.7682       4.2323
      1.5884      -1.2925     -0.29588
           0            0            0
           0            0            0
```

dal punto di vista di uno **Spazio Lineare**

Esempio MATLAB: PCA
e matrice di covarianza

$$\mathbf{X} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \text{campioni} \\ \begin{array}{ccc} X_1 & X_2 & X_3 \end{array} \\ \begin{bmatrix} -1 & -2 & 4 \\ 1 & 3 & 0 \\ 2 & 1 & 3 \\ 2 & 0 & -1 \end{bmatrix} \end{array}$$

variabili

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % numero dei campioni
mu = mean(X,2); % vettore colonna delle medie
```

```
[base,comp,lambda] = pca(X','Economy',false);
lambda
```

```
lambda =
       13.859
       2.1405
            0
            0
```

```
C = cov( X' );
[V,D] = eig(C,'vector');
[D,j]=sort(D,'descend'); V=V(:,j);
D
```

```
D =
       13.859
       2.1405
   3.0255e-15
  -7.4954e-16
```

```
Xc=X - repmat(mu,1,N);
Scat=Xc*Xc'; % scatter matrix
[Vscat,Dscat]=eig(Scat,'vector');
[Dscat,j]=sort(Dscat,'descend');
Vscat=Vscat(:,j);
all(Dscat == (N-1)*D)    autovalori scalati
ans = logical   1
all(all(Vscat == V))     stessi autovettori
ans = logical   1
```

```
V
V =
      -0.8633     -0.043656
       0.35242     -0.53471
      -0.25255       0.2328
       0.25833      0.81116
```

```
base
base =
       0.8633     -0.043656
      -0.35242     -0.53471
       0.25255       0.2328
      -0.25833      0.81116
```

```
 0.4782     -0.15534
 0.69417     0.32866
-0.13265     0.92974
 0.5214     -0.058544
```

```
0.34479      0.36596
0.76671      0.045205
0.33028     -0.87917
0.42918      0.30181
```

≠

```
rank([V(:,3:4) base(:,3:4)])
ans =
     2
```

linearm. dipendenti

dal punto di vista di uno **Spazio Lineare**

Esempio MATLAB: PCA e SVD (singular value decomposition)

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

$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\,\mathbf{S}_{m\times n}\,\mathbf{V}_{n\times n}^{T}$

reduced SVD: `[U,S,V]=svd(A,'econ')`

$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times r}\,\mathbf{S}_{r\times r}\,\mathbf{V}_{r\times n}^{T}$, rank(**A**)=r

$\mathbf{X}$

variabili	X_1	X_2	X_3
campioni			
x_1	−1	−2	4
x_2	1	3	0
x_3	2	1	3
x_4	2	0	−1

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % numero dei campioni
mu = mean(X,2); % vettore colonna delle medie
```

```
[base,comp,lambda] = pca(X','Economy',true);
lambda
```

```
lambda =
       13.859
       2.1405
```

```
Xc = X-repmat(mu,1,N);
[U,S,V] = svd(Xc','econ');
diag(S*S')/(N-1)
```

```
ans =
       13.859
       2.1405
    8.2173e-33
```

```
[U,S,V]=svd(Xc'/sqrt(N-1),'econ');
diag(S*S')
```

```
V
V =
     -0.8633      0.043656     -0.4782
      0.35242      0.53471     -0.69417
     -0.25255     -0.2328       0.13265
      0.25833     -0.81116     -0.5214
```

`svd(A,'econ')` A(m×n)

- m > n : solo le prime n colonne di U sono calcolate, e S è n×n.
- m = n : `svd(A,"econ")` è equivalente a `svd(A)`.
- m < n : solo le prime m colonne di V sono calcolate, e S è m×m.

```
base
base =
      0.8633      -0.043656
     -0.35242     -0.53471
      0.25255      0.2328
     -0.25833      0.81116
```

riduzione della dimensionalità

delle 4 dimensioni delle variabili casuali, solo due sono sufficienti a descrivere tutti i dati.

dal punto di vista di uno **Spazio Lineare**

Esempio MATLAB: PCA e SVD

$$\mathbf{X} = \begin{array}{c|ccc} & X_1 & X_2 & X_3 \\ \hline x_1 & -1 & -2 & 4 \\ x_2 & 1 & 3 & 0 \\ x_3 & 2 & 1 & 3 \\ x_4 & 2 & 0 & -1 \end{array}$$

(campioni; variabili)

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % numero dei campioni
mu = mean(X,2); % vettore colonna delle medie
```

```
[base,comp,lambda] = pca(X','Economy',false);
lambda
```

```
lambda =
       13.859
       2.1405
            0
            0
```

```
Xc = X-repmat(mu,1,N);
[U,S,V] = svd(Xc',0);
diag(S'*S)/(N-1)
```

```
ans =
       13.859
       2.1405
   8.2173e-33
            0
```

```
[U,S,V]=svd(Xc'/sqrt(N-1),0);
diag(S'*S)
```

svd(A,0) A(m×n)
m > n : svd(A,0) è equivalente a svd(A,"econ").
m ≤ n : svd(A,0) è equivalente a svd(A).

```
V
V =
     -0.8633      0.043656     -0.4782       0.15534
      0.35242     0.53471      -0.69417     -0.32866
     -0.25255    -0.2328        0.13265     -0.92974
      0.25833    -0.81116      -0.5214       0.058544
```

```
base
base =
      0.8633     -0.043656      0.4782     -0.15534
     -0.35242    -0.53471       0.69417     0.32866
      0.25255     0.2328       -0.13265     0.92974
     -0.25833     0.81116       0.5214     -0.058544
```

riduzione della dimensionalità
delle 4 dimensioni delle variabili casuali, solo due sono sufficienti a descrivere tutti i dati.

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % numero dei campioni
mu = mean(X,2); % vettore colonna delle medie
```

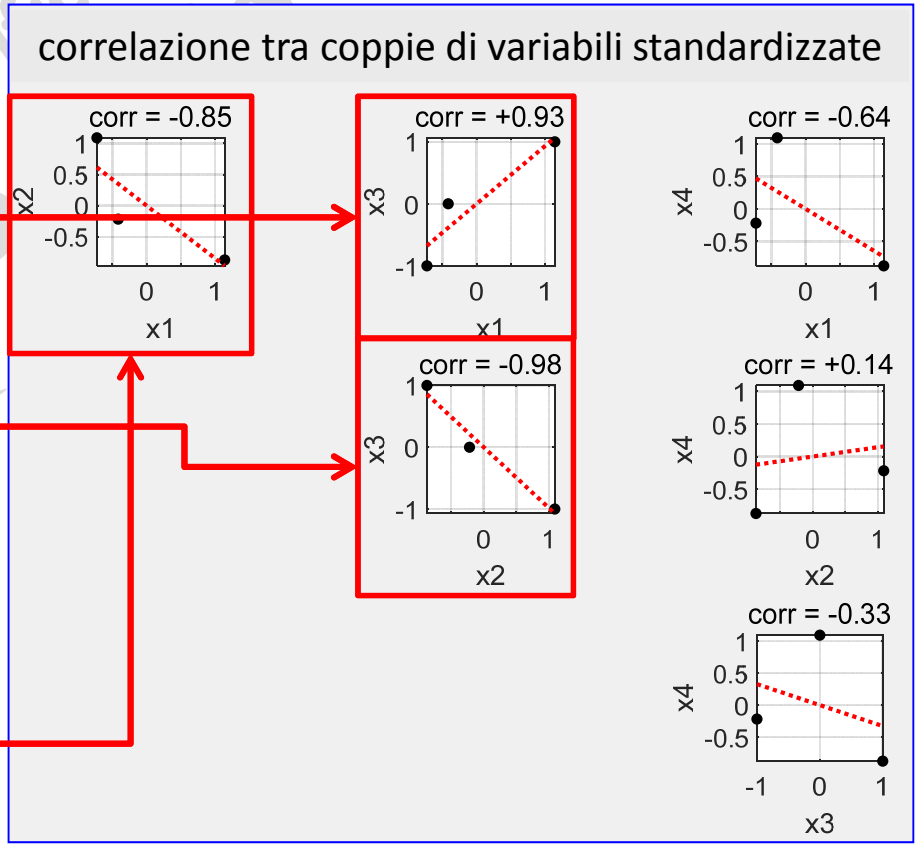
```
R=corrcoef(X')  % matrice di correlazione
```

R = %	x1	x2	x3	x4
x1	1	-0.84856	0.93326	-0.6449
x2	-0.84856	1	-0.98198	0.14286
x3	0.93326	-0.98198	1	-0.32733
x4	-0.6449	0.14286	-0.32733	1

```
Xs = normalize(X'); % matrice standardiz.
```

Alta correlazione tra:

- 2ª e 3ª variabile → correlazione negativa
- 1ª e 3ª variabile → correlazione positiva
- 1ª e 2ª variabile → correlazione negativa



Riassunto

dal punto di vista **statistico**

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1];
N=size(X,1); % numero di campioni
mu=mean(X);  % vettore riga delle medie
Xs=normalize(X);        % matrice standardizz.
Xc=X-repmat(mu,N,1); % matrice centrata
[Bx,Cx,Lx]=pca(X,'Economy',false);
[Bc,Cc,Lc]=pca(Xc,'Economy',false);
[Bs,Cs,Ls]=pca(Xs,'Economy',false);
```

dal punto di vista di uno **Spazio Lineare**

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N=size(X,2);  % numero di campioni
mu=mean(X,2); % vettore col di medie
Xs=normalize(X');      % matrice standardizz.
Xc=X-repmat(mu,1,N); % matrice centrata
[Bx,Cx,Lx]=pca(X','Economy',false);
[Bc,Cc,Lc]=pca(Xc','Economy',false);
[Bs,Cs,Ls]=pca(Xs,'Economy',false);
```

```
Lx % Lx == Lc
Lx =
      13.859
      2.1405
           0
           0
Bx % Bx == Bc
Bx =
    0.8633    -0.043656      0.4782     -0.15534
  -0.35242     -0.53471     0.69417      0.32866
   0.25255       0.2328    -0.13265      0.92974
  -0.25833      0.81116      0.5214    -0.058544
Cx % Cx == Cc
Cx =
   -1.4641       1.5884            0            0
   -2.7682      -1.2925            0            0
    4.2323     -0.29588            0            0
```

≠

```
Ls
Ls =
       3.0482
      0.95179
            0
            0
Bs
Bs =
    0.5671     -0.14387        0.775      0.23892
  -0.52375     -0.41489      0.07813       0.7399
   0.55812      0.23029       -0.545      0.58176
  -0.30428      0.86841      0.31024       0.2388
Cs
Cs =
  -0.45293       1.0977            0            0
   -1.4748     -0.76804            0            0
    1.9277     -0.32968            0            0
```

```
disp(rank([Bx(:,1:2)  Bs(:,1:2)]))
     4
```

ACS parte 2: ACS_09b

Argomenti trattati

- **Cos'è la PCA?**
- **Derivazione dell'algoritmo PCA.**
- **Algoritmi PCA.**
- **Interpretazione geometrica della PCA.**

Analisi delle Componenti Principali

Principal Component Analysis (PCA) appare in molti campi della matematica applicata con nomi diversi (Trasformata di Karhunen-Loeve, Trasformata di Hotelling, …).

La **PCA** è stata creata nel 1901 da Karl Pearson, ma Harold Hotelling l'ha sviluppata indipendentemente nel 1933 e le ha dato il nome di "Principal Component Analysis method".

Cos'è la PCA?

La **PCA** è una procedura che trasforma variabili (possibilmente) correlate in variabili incorrelate dette componenti principali.

Per cosa è usata la PCA?

Per la riduzione della dimensionalità dei dati ("dimensionality reduction"), cioè per ridurre il size dei dati senza (quasi) perdita di informazioni.

Come funziona la PCA?

La PCA cerca le direzioni con massima variazione dei dati.

La 1ª direzione principale è individuata da un autovettore relativo al massimo autovalore della Matrice di Covarianza dei dati.
Poi analogamente la 2ª direzione principale ...

Inoltre, la PCA proietta i dati su queste direzioni, ottenendo le loro componenti principali.

Per derivare la PCA

Richiami:

Proprietà del gradiente (∇) del prodotto scalare standard

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{d} x_i y_i$$

$\langle \mathbf{x}, v \rangle = \mathbf{x}^T v = v^T \mathbf{x}$

1. $F(x_1, x_2, \ldots, x_d) = \langle \mathbf{x}, v \rangle = \sum_{i=1}^{d} x_i v_i \Rightarrow \nabla_{\mathbf{x}} F = v$

v non dipende da $\mathbf{x}$

simile a: $f(x)=\alpha x \Rightarrow f'(x)=\alpha$

Dim.:

j^a componente del gradiente $\frac{\partial}{\partial x_j} F = v_j$

$\langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{x}^T \mathbf{x}$

2. $F(x_1, x_2, \ldots, x_d) = \langle \mathbf{x}, \mathbf{x} \rangle = \|\mathbf{x}\|_2^2 = \sum_{i=1}^{d} x_i^2 \Rightarrow \nabla_{\mathbf{x}} F = 2\mathbf{x}$

simile a: $f(x)=x^2 \Rightarrow f'(x)=2x$

Dim.:

j^a componente del gradiente $\frac{\partial}{\partial x_j} F = 2x_j$

$\langle \mathbf{x}, A\mathbf{x} \rangle = \mathbf{x}^T A\mathbf{x} = (A\mathbf{x})^T \mathbf{x} = \mathbf{x}^T A\mathbf{x}$

3. $F(x_1, x_2, \ldots, x_d) = \langle \mathbf{x}, \mathbf{A}\mathbf{x} \rangle = \sum_{i=1}^{d} x_i \left[\sum_{j=1}^{d} A_{ij} x_j \right] \Rightarrow \nabla_{\mathbf{x}} F = 2\mathbf{A}\mathbf{x}$

$\mathbf{A}$ è simmetrica e non dipende da $\mathbf{x}$

Per derivare la PCA

Richiami:

Proprietà del gradiente (∇) del prodotto scalare standard

3. $F(x_1, x_2, \ldots, x_d) = \langle \mathbf{x}, \mathbf{Ax} \rangle = \sum_{i=1}^{d} x_i \left[\sum_{k=1}^{d} A_{ik} x_k \right] \Rightarrow \nabla_{\mathbf{x}} F = 2\mathbf{Ax}$

A è simmetrica e non dipende da **x**

Dim.:

$$\mathbf{y}(\mathbf{x}) = \mathbf{Ax} \quad \Leftrightarrow \quad y_i(\mathbf{x}) = \sum_{k=1}^{d} A_{ik} x_k \quad \Rightarrow \quad \sum_{i=1}^{d} x_i y_i(\mathbf{x}) = F(x_1, \ldots, x_d)$$

$(\nabla_{\mathbf{x}} F)_j$ = j^a componente del gradiente di $F()$:

$$\frac{\partial}{\partial x_j} F = \sum_{i=1}^{d} \frac{\partial}{\partial x_j} \{x_i y_i(\mathbf{x})\} = y_j(\mathbf{x}) + \sum_{i=1}^{d} x_i \frac{\partial}{\partial x_j} \{y_i(\mathbf{x})\} =$$

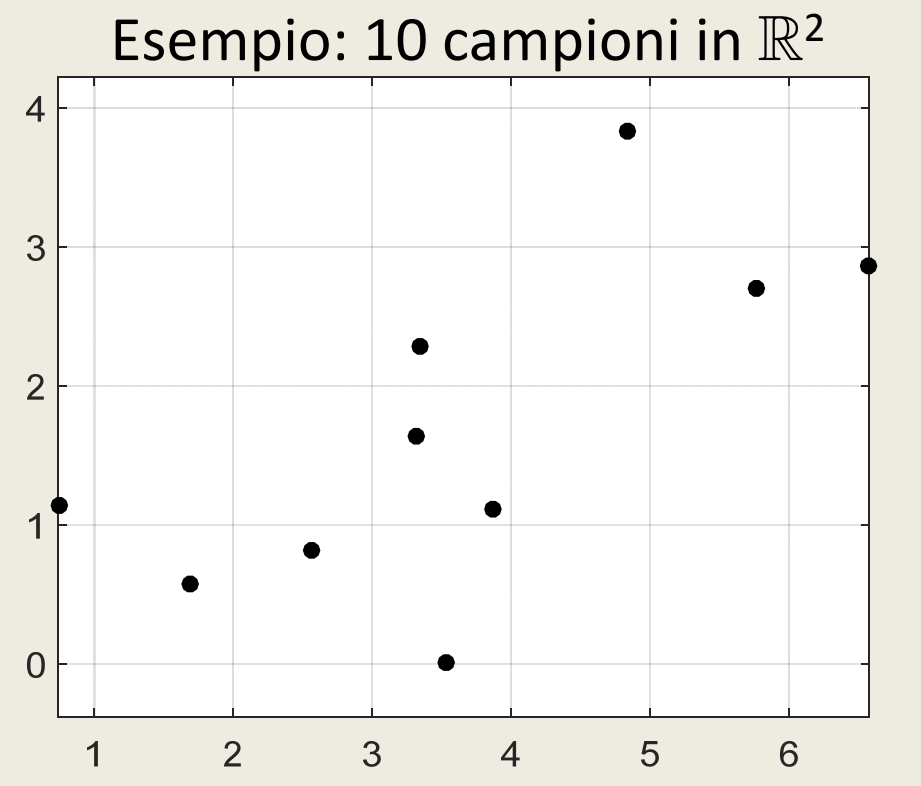
$$= y_j(\mathbf{x}) + \sum_{i=1}^{d} x_i \frac{\partial}{\partial x_j} \sum_{k=1}^{d} A_{ik} x_k = y_j(\mathbf{x}) + \sum_{i=1}^{d} x_i \sum_{k=1}^{d} \frac{\partial}{\partial x_j} \{A_{ik} x_k\} =$$

$$= y_j(\mathbf{x}) + \sum_{i=1}^{d} A_{ij} x_i = y_j(\mathbf{x}) + \sum_{i=1}^{d} A_{ji} x_i = y_j(\mathbf{x}) + y_j(\mathbf{x}) = 2y_j(\mathbf{x})$$

poiché **A** è simmetrica

derivazione della PCA

Si considerino N campioni $\mathbf{x}^{[k]} \in \mathbb{R}^d$.



Si cerca un sottospazio dello *Spazio Affine* $\mathbb{R}^d$, di dimensione $K << d$, che descriva significativamente i dati.
Il sottospazio è detto sottospazio PCA "ridotto".

derivazione della PCA (cont.)

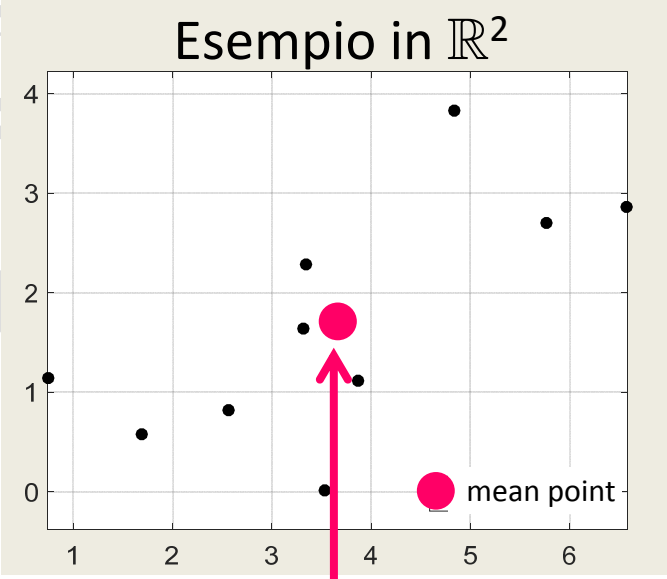
Sottospazio PCA di dimensione 0 (un singolo punto): si vuole descrivere i dati mediante un solo punto **c**, tale che "esso minimizzi la somma delle sue distanze euclidee da tutti i campioni", cioè

Trovare **c** che minimizza J_0 : $J_0 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2$

somma di quadrati di norma-2

⇩

funzione convessa



c deve essere la media campionaria **m**.

derivazione della PCA (cont.)

Il punto c che **minimizza** $J_0 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2$ è la media campionaria m : Perché?

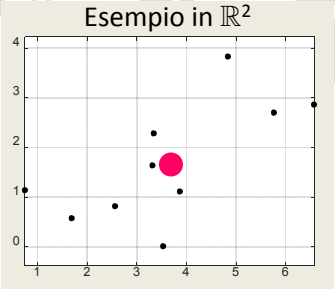
Dimostrazione:

Dalle proprietà del gradiente del prodotto scalare standard e della norma indotta:

$$0 \leq J_0 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} \right\|_2^2 + N \left\| \mathbf{c} \right\|_2^2 - 2 \sum_{k=1}^{N} \left\langle \mathbf{x}^{[k]}, \mathbf{c} \right\rangle$$

$$\nabla_{\mathbf{c}} J_0 = 2N\mathbf{c} - 2 \sum_{k=1}^{N} \nabla_{\mathbf{c}} \left\langle \mathbf{x}^{[k]}, \mathbf{c} \right\rangle = 2N\mathbf{c} - 2 \sum_{k=1}^{N} \mathbf{x}^{[k]} =$$

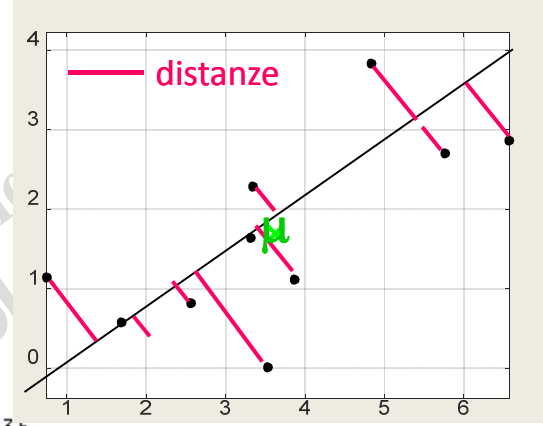
$$= 2 \left(N\mathbf{c} - \sum_{k=1}^{N} \mathbf{x}^{[k]} \right) = 0 \quad \Leftrightarrow \quad \boxed{\mathbf{c} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}^{[k]}}$$

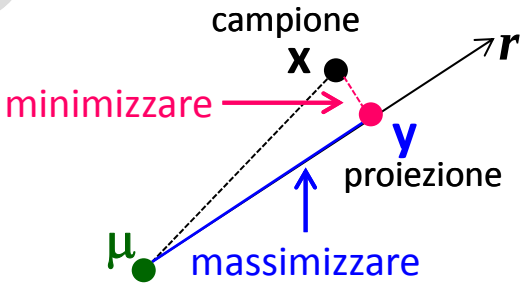


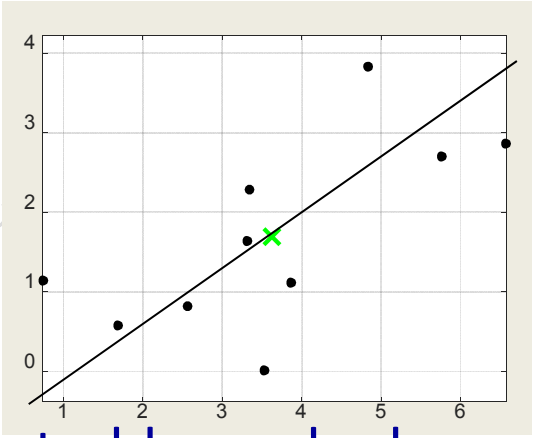
Tuttavia, la media campionaria m da sola non è in grado di descrivere sufficientemente tutti i dati.

derivazione della PCA Incrementale

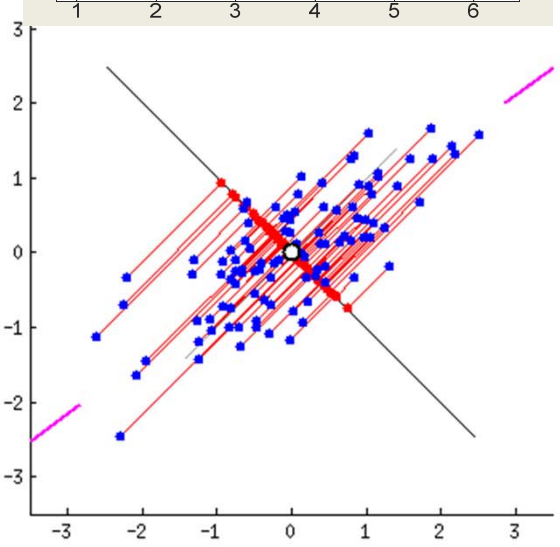
Allora, si cerca una **retta *r*** (sottospazio PCA di dimensione 1) tale che "essa minimizzi la somma delle sue distanze euclidee da tutti i campioni", cioè che minimizzi la somma dei residui delle proiezioni.

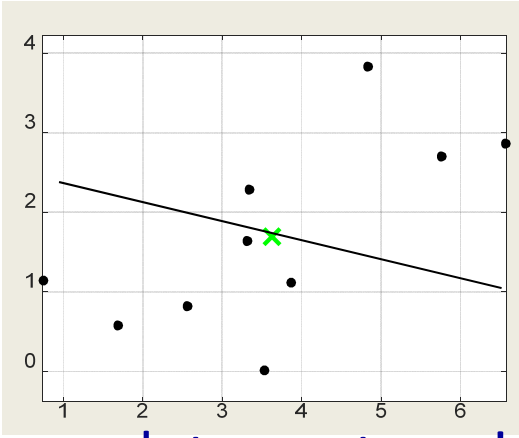






potrebbe andar bene ...





assolutamente no!

derivazione della PCA Increm. (cont.)

Per trovare la 1ª *direzione principale*, si cerca una retta r, nello spazio affine $\mathbb{R}^d$, passante per un punto $\mathbf{c}$ e parallela ad un vettore normalizzato $\mathbf{e}$ ($\|\mathbf{e}\|_2=1$):

$$r : \mathbf{y} = \mathbf{c} + \rho\,\mathbf{e}, \quad \rho \in \mathbb{R}$$

tale che "r minimizzi la somma delle distanze euclidee tra tutti i campioni $\mathbf{x}^{[k]}$ e le loro proiezioni ortogonali $\mathbf{y}^{[k]}$ sulla retta".

matrice di proiezione (sullo spazio direttore)

$$\mathbf{P} : \mathbf{v}^{[k]} = \mathbf{x}^{[k]} - \mathbf{c} \longrightarrow \mathbf{w}^{[k]} = \mathbf{P}\,\mathbf{v}^{[k]} = \mathbf{y}^{[k]} - \mathbf{c} = \alpha^{[k]}\,\mathbf{e}$$

P agisce sullo Spazio Lineare dei vettori che partono da **c** (come Origine)

Trovare $\mathbf{c}$, $\mathbf{e}$ e $\alpha^{[k]}$ tali da

minimizzare $$J_1 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \mathbf{y}^{[k]} \right\|_2^2$$

Errore quadratico di ricostruzione
(SSE: sum of squared errors)

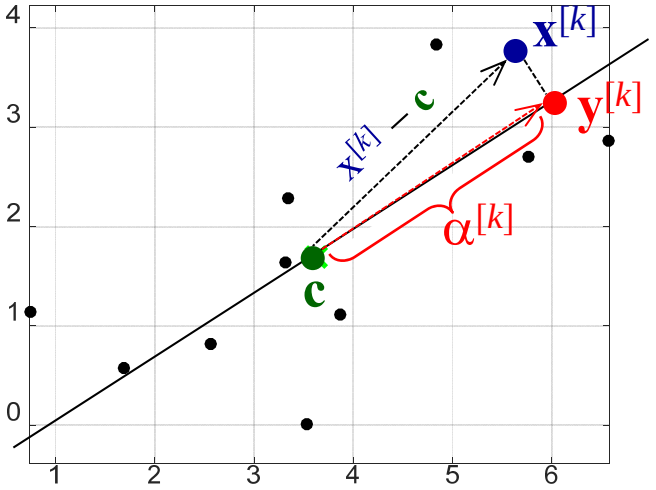
derivazione della PCA Increm. (cont.)

$(\mathbf{y}^{[k]}-\mathbf{c})$ è la proiezione ortogonale di $(\mathbf{x}^{[k]}-\mathbf{c})$ sulla retta $r \,||\, \mathbf{e}$

$$\mathbf{P} : (\mathbf{x}^{[k]} - \mathbf{c}) \longrightarrow (\mathbf{y}^{[k]} - \mathbf{c}) = \mathbf{P}(\mathbf{x}^{[k]} - \mathbf{c}) = \alpha^{[k]}\,\mathbf{e}$$

Per ottenere questi vettori, tutti i punti sono stati centrati in c

minimizzare $J_1\left(\mathbf{c}, \alpha^{[k]}, \mathbf{e}\right) = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \underbrace{\left(\mathbf{c} + \alpha^{[k]}\mathbf{e}\right)}_{\mathbf{y}^{[k]}} \right\|_2^2$



Per minimizzare J_1, lo scalare $\alpha^{[k]}$ deve essere la componente del vettore $(\mathbf{x}^{[k]} - \mathbf{c})$ lungo r, cioè:

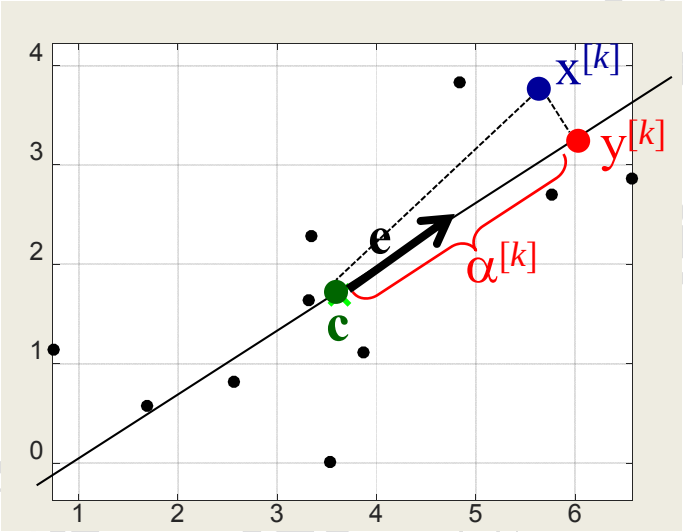
$$\alpha^{[k]} = \langle (\mathbf{x}^{[k]} - \mathbf{c}), \mathbf{e} \rangle = (\mathbf{x}^{[k]} - \mathbf{c})^T \mathbf{e} = \langle \mathbf{e}, (\mathbf{x}^{[k]} - \mathbf{c}) \rangle = \mathbf{e}^T (\mathbf{x}^{[k]} - \mathbf{c})$$

dove $\langle \cdot, \cdot \rangle$ è il prodotto scalare standard che induce la norma euclidea.

derivazione della PCA Increm. (cont.)

minimizzare $J_1\left(\mathbf{c},\alpha^{[k]},\mathbf{e}\right)=\sum_{k=1}^{N}\left\|\mathbf{x}^{[k]}-\left(\mathbf{c}+\alpha^{[k]}\mathbf{e}\right)\right\|_2^2$

Per ipotesi: $\|\mathbf{e}\|_2=1$



Applicando le proprietà del gradiente del prodotto scalare standard e della norma indotta, si ha:

Espanderlo come il quadrato di un binomio $(A-B)^2 = A^2 + B^2 - 2AB$

$$\left\|\left(\mathbf{x}^{[k]}-\mathbf{c}\right)-\alpha^{[k]}\mathbf{e}\right\|_2^2=\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2+\left(\alpha^{[k]}\right)^2\|\mathbf{e}\|_2^2-2\alpha^{[k]}\left\langle\left(\mathbf{x}^{[k]}-\mathbf{c}\right),\mathbf{e}\right\rangle=$$

$$=\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2+\left(\alpha^{[k]}\right)^2-2\left(\alpha^{[k]}\right)^2=\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2-\left(\alpha^{[k]}\right)^2=$$

$$=\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2-\mathbf{e}^{\mathsf{T}}\left(\mathbf{x}^{[k]}-\mathbf{c}\right)\left(\mathbf{x}^{[k]}-\mathbf{c}\right)^{\mathsf{T}}\mathbf{e}$$

derivazione della PCA Increm. (cont.)

Sostituendo il risultato in J_1, si ha:

$$J_1\left(\mathbf{c},\alpha^{[k]},\mathbf{e}\right)=\sum_{k=1}^{N}\left\|\mathbf{x}^{[k]}-\left(\mathbf{c}+\alpha^{[k]}\mathbf{e}\right)\right\|_2^2=$$

$$=\sum_{k=1}^{N}\left\{\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2-\mathbf{e}^{\mathsf{T}}\left(\mathbf{x}^{[k]}-\mathbf{c}\right)\left(\mathbf{x}^{[k]}-\mathbf{c}\right)^{\mathsf{T}}\mathbf{e}\right\}=$$

$$=\sum_{k=1}^{N}\left\{\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2\right\}-\mathbf{e}^{\mathsf{T}}\left\{\sum_{k=1}^{N}\left[\left(\mathbf{x}^{[k]}-\mathbf{c}\right)\left(\mathbf{x}^{[k]}-\mathbf{c}\right)^{\mathsf{T}}\right]\right\}\mathbf{e}$$

minimizzare ⟺ minimizzare + massimizzare

derivazione della PCA Increm. (cont.)

$$J_1(\mathbf{c},\mathbf{e}) = \sum_{k=1}^{N} \left\{ \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2 \right\} - \mathbf{e}^{\mathsf{T}} \left\{ \sum_{k=1}^{N} \left[\left(\mathbf{x}^{[k]} - \mathbf{c} \right) \left(\mathbf{x}^{[k]} - \mathbf{c} \right)^{\mathsf{T}} \right] \right\} \mathbf{e}$$

c=m $\Rightarrow$ questa è la **scatter matrix** S

$$S = \sum_{k=1}^{N} \left[\left(\mathbf{x}^{[k]} - \mathbf{m} \right) \left(\mathbf{x}^{[k]} - \mathbf{m} \right)^{\mathsf{T}} \right]$$

S è proporzionale alla matrice di covarianza dei dati $\mathbf{x}^{[k]}$.

minimizzare

minimizzare

maximizzare

c deve essere la *media campionaria* **m** dei dati $\mathbf{x}^{[k]}$

e è la direzione che massimizza la varianza dei dati

derivazione della PCA Increm. (cont.)

Problema di Ottimizzazione vincolata

S scatter matrix

Minimizzare: $J_1(\mathbf{c},\mathbf{e}) = \sum_{k=1}^{N} \left\{ \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2 \right\} - \mathbf{e}^T \mathrm{S}\, \mathbf{e}$

soggetto al vincolo di eguaglianza: $\mathbf{e}^T\mathbf{e} = 1$.

Un problema di ottimizzazione vincolata con soli vincoli di eguaglianza può essere risolto mediante il **metodo dei moltiplicatori di Lagrange**, in base al quale si cercano i punti stazionari della funzione Lagrangiana $\mathbb{L}$:

$$\mathbb{L}(\mathbf{c},\mathbf{e},\lambda) = J_1(\mathbf{c},\mathbf{e}) + \lambda(\mathbf{e}^T\mathbf{e} - 1), \quad \lambda \in \mathbb{R}$$

Questi sono gli zeri del gradiente ∇ di $\mathbb{L}$().

derivazione della PCA Increm. (cont.)

minimo di $\mathbb{L}(\mathbf{c},\mathbf{e},\lambda)=\sum_{k=1}^{N}\left\{\left\|\mathbf{x}^{[k]}-\mathbf{c}\right\|_2^2\right\}-\mathbf{e}^{\mathsf{T}}S\,\mathbf{e}+\lambda\left(\mathbf{e}^{\mathsf{T}}\mathbf{e}-1\right)$ $\Longleftrightarrow$ $\nabla\mathbb{L}=0$

$\frac{\partial}{\partial\lambda}\mathbb{L}(\mathbf{c},\mathbf{e},\lambda)=0$ $\Longleftrightarrow$ $\mathbf{e}^{\mathsf{T}}\mathbf{e}=1$ (vero per ipotesi)

$\frac{\partial}{\partial\mathbf{c}}\mathbb{L}(\mathbf{c},\mathbf{e},\lambda)=0$ $\Longleftrightarrow$ $\mathbf{c}=\mathbf{m}$ (media campionaria)

$\frac{\partial}{\partial\mathbf{e}}\mathbb{L}(\mathbf{c},\mathbf{e},\lambda)=-2S\mathbf{e}+2\lambda\mathbf{e}=-2\boxed{(S\mathbf{e}-\lambda\mathbf{e})=0}$

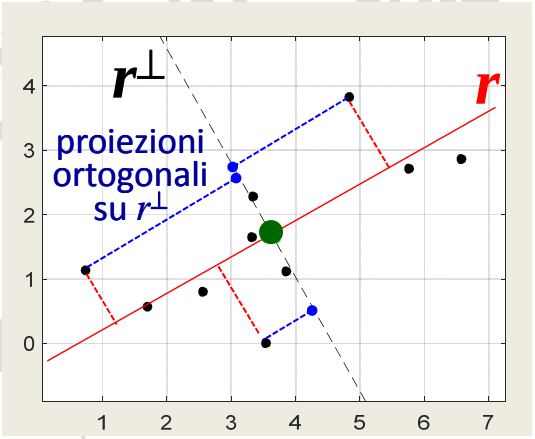
$\Longleftrightarrow$ $\boxed{S\mathbf{e}=\lambda\mathbf{e}}$ (tipica eq. autovalori/autovettori) cioè:

- **e** è un autovettore relativo all'autovalore λ della *Scatter matrix* S (**e** è anche un autovettore della *matrice di Covarianza*);
- λ è il massimo autovalore, perché $\min J_1$ equivale a:

  $\min\{-\mathbf{e}^{\mathsf{T}}S\mathbf{e}\} = \max\{\mathbf{e}^{\mathsf{T}}S\mathbf{e}\} = \max\{\mathbf{e}^{\mathsf{T}}\lambda\mathbf{e}\} = \max\{\lambda\mathbf{e}^{\mathsf{T}}\mathbf{e}\} = \max\{\lambda\}$

Algoritmo PCA incrementale

Nello spazio lineare (centrato in **m**), dopo aver trovato la prima direzione principale (r : r=span{**e**}), per cercare la seconda direzione principale, si considera $r^{\perp}$ (il complemento ortogonale di r) e si ripete la precedente procedura sulle proiezioni ortogonali dei dati $\mathbf{x}^{[k]}$ su $r^{\perp}$.

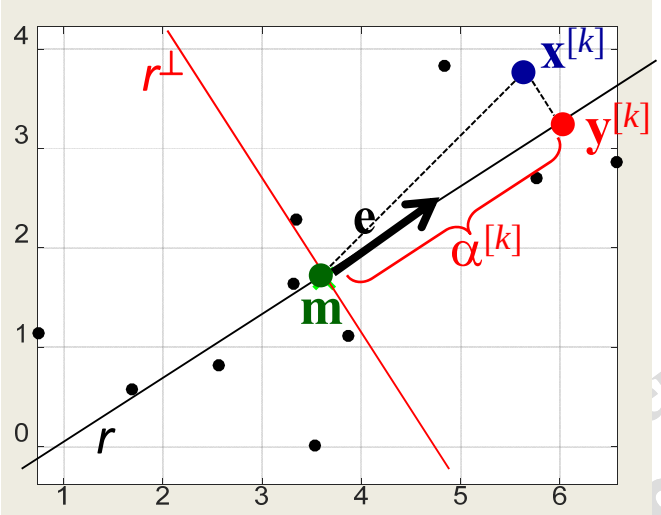


Ricorda che $\mathbb{R}^d = r \oplus r^{\perp}$ e $r \perp r^{\perp}$: allora $r^{\perp}$ è un sottospazio di dimensione d−1. La procedura può essere ripetuta fino a d=1.

Come proiettare ortogonalmente su $r^{\perp}$?

Come proiettare ortogonalmente su $r^{\perp}$?

$(\mathbf{y}^{[k]}-\mathbf{m})$ è la proiezione ortogonale di $(\mathbf{x}^{[k]}-\mathbf{m})$ su r



$$\mathbf{P} : (\mathbf{x}^{[k]} - \mathbf{m}) \longrightarrow (\mathbf{y}^{[k]} - \mathbf{m}) = \alpha^{[k]}\mathbf{e}$$

$\mathbf{P}$ è la *matrice di proiezione ortogonale* su r=span{$\mathbf{e}$} ($\mathbf{e}$: base ortonormale):

formula semplificata $\Rightarrow$ $\mathbf{P} = \mathbf{e}\,\mathbf{e}^{\mathsf{T}}$

$$(\mathbf{y}^{[k]} - \mathbf{m}) = \mathbf{P}(\mathbf{x}^{[k]} - \mathbf{m}) = \mathbf{e}\,\overbrace{\mathbf{e}^{\mathsf{T}}(\mathbf{x}^{[k]} - \mathbf{m})}^{\alpha^{[k]}}$$

$\Rightarrow$ $\mathrm{I} - \mathrm{P}$ è la matrice di proiezione ortogonale su $r^{\perp}$

$\mathrm{I} - \mathrm{P}$ è detta "matrice di proiezione complementare"

Dim.: Infatti, il vettore $\mathbf{x}^{[k]} - \mathbf{y}^{[k]}$, ortogonale a r, può essere scritto come:

$$(\mathbf{x}^{[k]} - \mathbf{m}) - (\mathbf{y}^{[k]} - \mathbf{m}) = (\mathbf{x}^{[k]} - \mathbf{m}) - \mathrm{P}(\mathbf{x}^{[k]} - \mathbf{m}) = (\mathrm{I}-\mathrm{P})(\mathbf{x}^{[k]} - \mathbf{m})$$

Esempio: PCA in MATLAB

Statistics and Machine Learning Toolbox

Campione random da Distribuzione Normale Multivariata

```
N=10; X=mvnrnd([3 1 1],[1 .2 .7; .2 1 0; .7 0 1],N); % mvnrnd(μ,Σ,N)
```

```
[basis,comp,lambda]=pca(X,'Economy',false);
```

base di $\mathbb{R}^d$

componenti principali

autovalori di cov(X)

v.c.

campioni

matrice dei dati **X**

dal punto di vista **statistico**

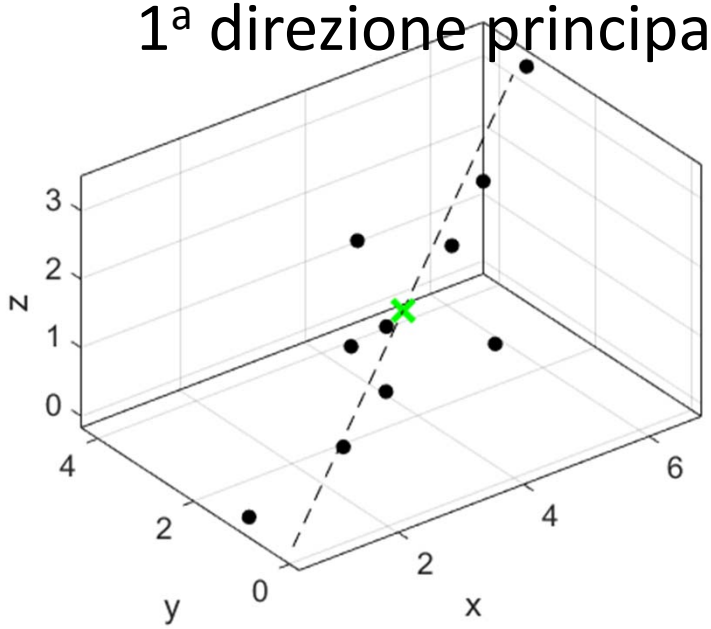
X di size (N×d) dove:

N è il numero dei campioni,

d è la dim. dello spazio dei dati

`'Economy',false`: per avere una base completa di $\mathbb{R}^d$

1ª direzione principale



base PCA di $\mathbb{R}^3$

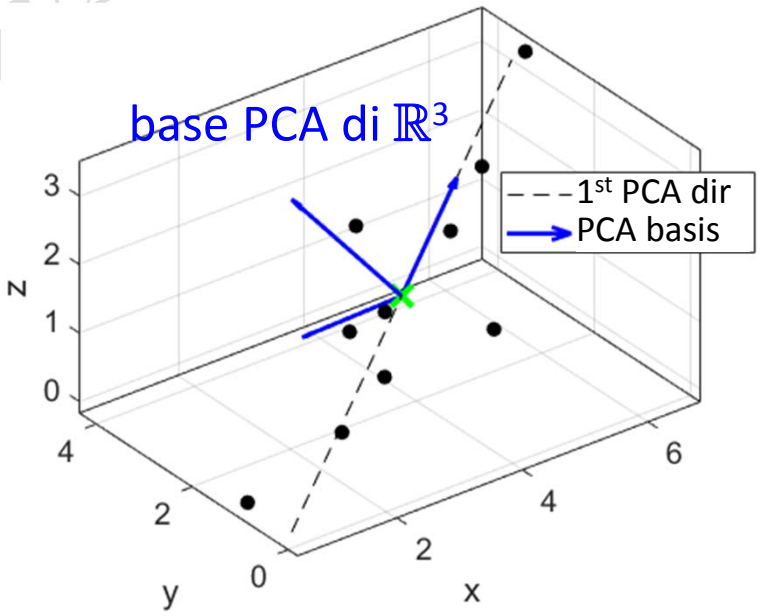
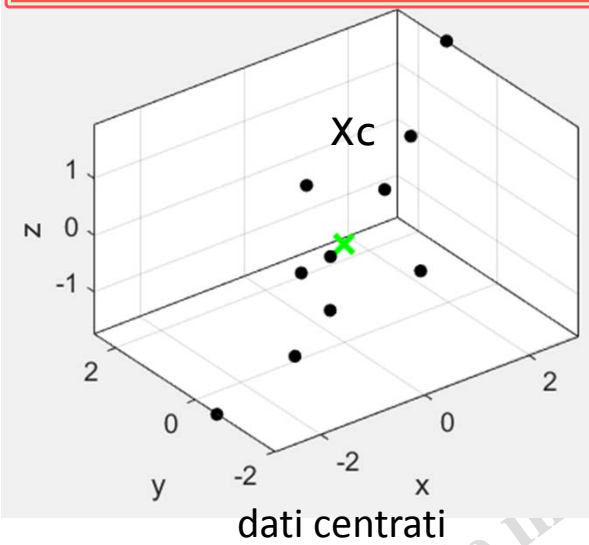


Illustrazione dell'algoritmo PCA incrementale (1)

```
N=10; X=mvnrnd([3 1 1],[1 .2 .7; .2 1 0; .7 0 1],N); % mvnrnd(μ,Σ,N)
mu=mean(X);   Xc=X-repmat(mu,size(X,1),1);
```

dal punto di vista **statistico**



dati centrati

base di *r* (**e**)

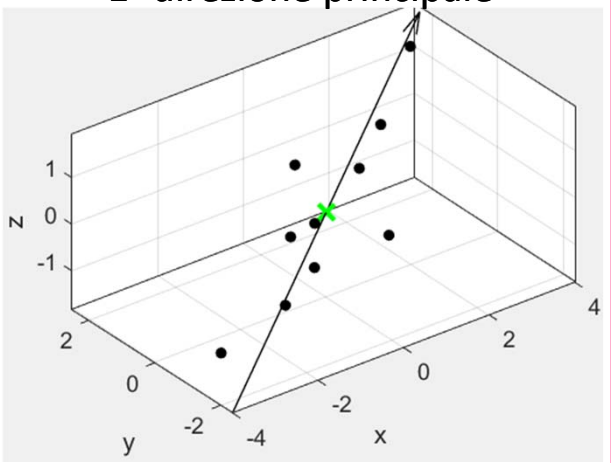
```
[base1,comp1,lambda1] = pca(X,'NumComponents',1);
```

```
[L1,w1] = powerMethod(Xc'*Xc/(N-1));
```

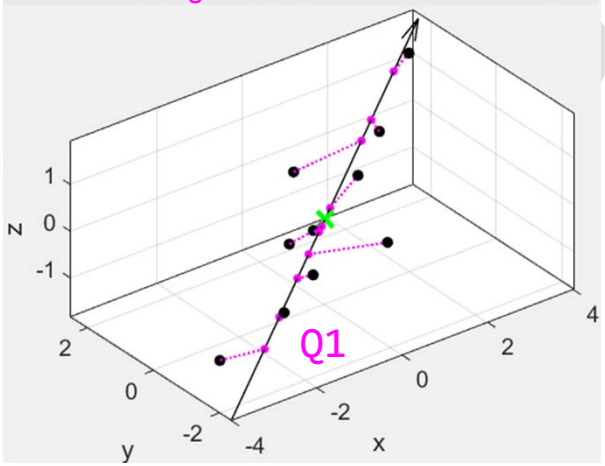
metodo potenza: trova il massimo autovalore e relativo autovettore

```
function [L1,w1]=powerMethod(A)
  nIter=200; N=size(A,2);
  w1=ones(N,1); w1=w1/norm(w1);
  for k=1:nIter % iterazioni
    w0=w1; w1=A*w0; w1=w1/norm(w1);
    L1=dot(w1,A*w1);
    absErr=norm(w1-w0);
    if absErr < 1e-10
        break % ferma le iterazioni
    end
  end
end
```

1ª direzione principale



Proiezioni ortogonali dei dati sulla retta PCA1



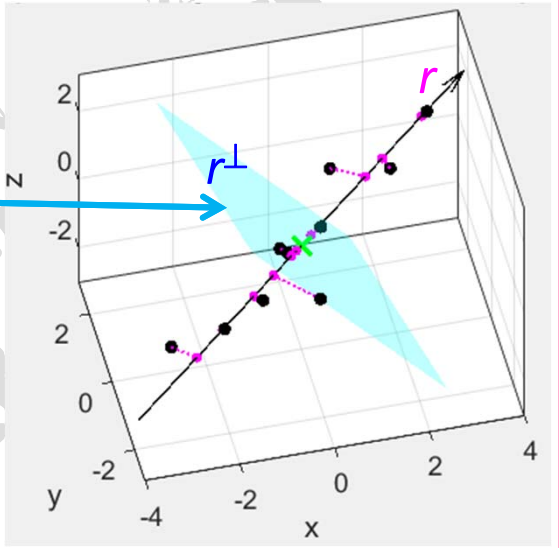

```
lambda1(1)
       4.7561
base1 =
       0.80209
       0.47385
        0.3635
```

```
L1 =
       4.7561
w1 =
       0.80209
       0.47385
        0.3635
```

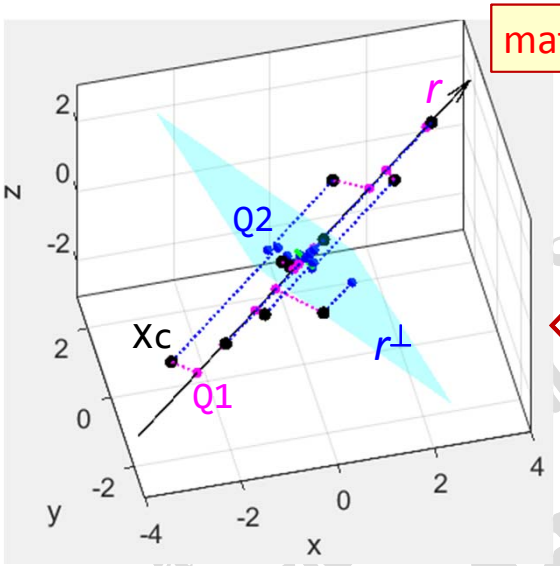
```
Pperp=base1*base1';  % matrice di proiezione ortogonale sulla retta
Q1=Pperp*Xc'; Q1=Q1';  % comp1 == Q1*base1
plot3(Q1(:,1),Q1(:,2),Q1(:,3),'m.','MarkerSize',18)
title('Proiezioni ortogonali dei dati sulla retta PCA1')
```

Illustrazione dell'algoritmo PCA incrementale (2)

```
B2=null(base1'); % base del complemento ortogonale di r
syms a b real;  plane=B2*[a;b];
h=ezsurf(plane(1),plane(2),plane(3),[-3 3]);
title('complemento ortogonale della 1^{a} direzione principale')
```



matrice di proiezione complementare I–P




```
Pcmpl=eye(3) - Pperp;
Q2=Pcmpl*Xc'; Q2=Q2'; % proiezioni su r^\perp
plot3(Q2(:,1),Q2(:,2),Q2(:,3),'b.','MarkerSize',18)
title('Proiezioni sul complemento ortogonale di r')
```

Componenti, risp. alla base del piano, dei vettori proiettati

```
Q3=B2'*Q2'; % componenti 2D risp. a B2 (equivalente a  Q3=B2\Q2')
Q3=Q3';
plot(Q3(:,1),Q3(:,2),'b.','MarkerSize',24); hold on
h=plot(0,0,'Xg'); set(h,'LineWidth',3,'MarkerSize',14)
axis equal; grid on; box on
set(gca,'FontSize',14); xlabel('x'); ylabel('y')
title('Proiezioni sul complemento ortogonale di r')
```

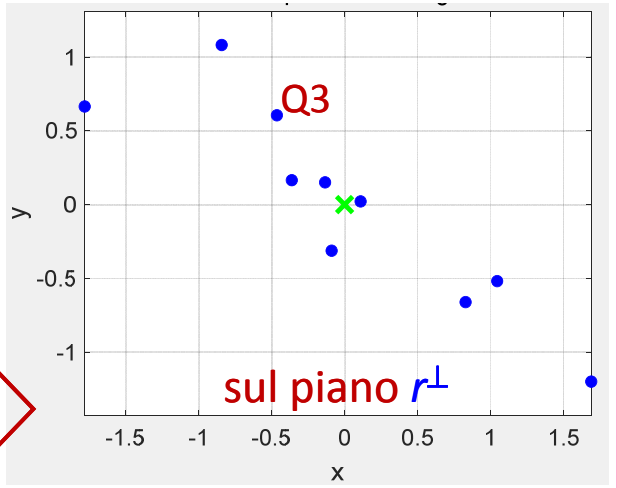
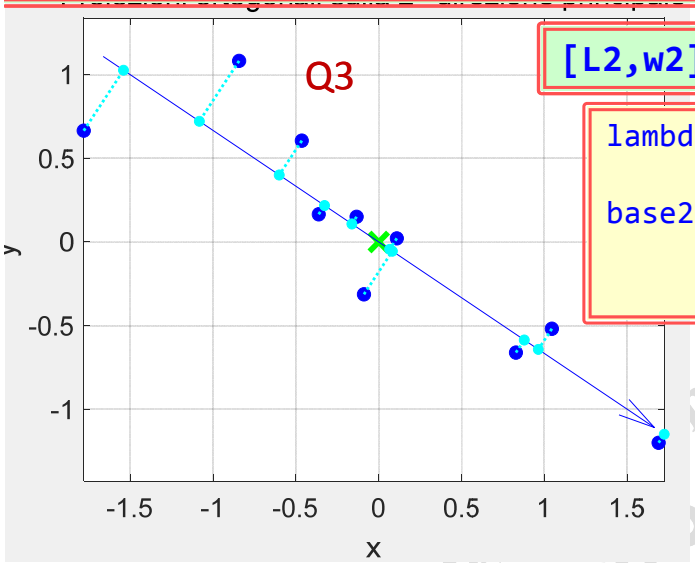


Illustrazione dell'algoritmo PCA incrementale (3)

PCA applicata ai dati 2D proiettati

```
[base2,comp2,lambda2]=pca(Q3,'NumComponents',1);
```

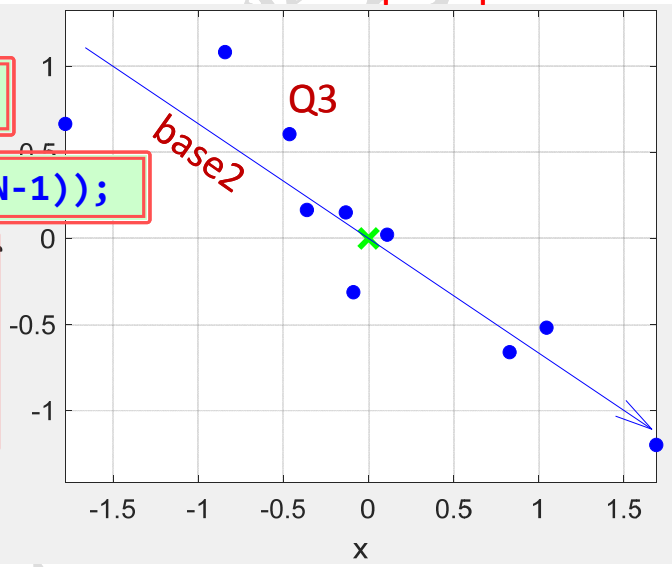



```
[L2,w2]=powerMethod(Q3'*Q3/(N-1));
```

```
lambda2(1)
      1.4005
base2 =
      0.83237
     -0.55422
```

```
L2 =
      1.4005
w2 =
      0.83237
     -0.55422
```

2ª direzione principale



confronto con PCA completa

```
[base,comp,lambda]=pca(X,'Economy',false);
```

```
comp1 =
    -0.85923
     1.9632
    -3.2115
    0.28882
   -0.30638
    -2.4193
    -1.4579
   -0.16089
     3.6831
     2.4801
```

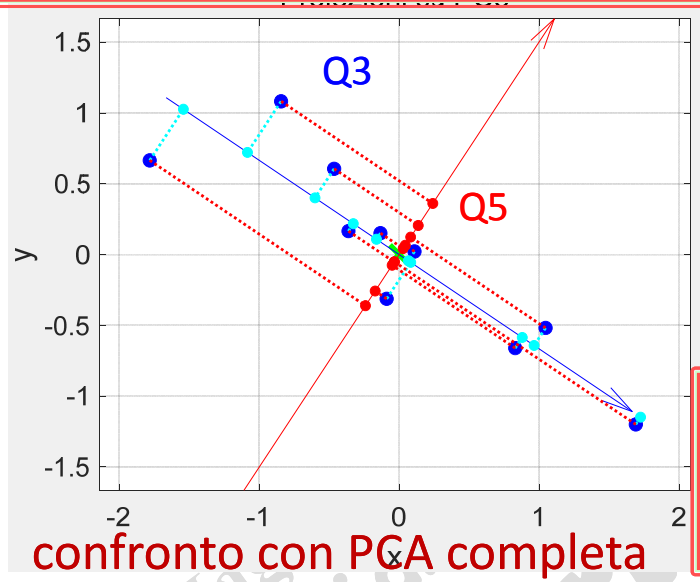
```
comp2 =
     -1.8518
      2.072
      1.158
     -1.3018
    0.079027
    -0.19491
    -0.39336
      1.0563
    -0.72228
    0.098876
```

```
comp =
    -0.85923     -1.8518     -0.4339
     1.9632       2.072    -0.061501
    -3.2115       1.158      0.14823
    0.28882     -1.3018      0.43342
   -0.30638    0.079027     0.078779
    -2.4193    -0.19491     0.051441
    -1.4579    -0.39336    -0.063018
   -0.16089      1.0563    -0.090606
     3.6831    -0.72228       0.2467
     2.4801    0.098876     -0.30955
```

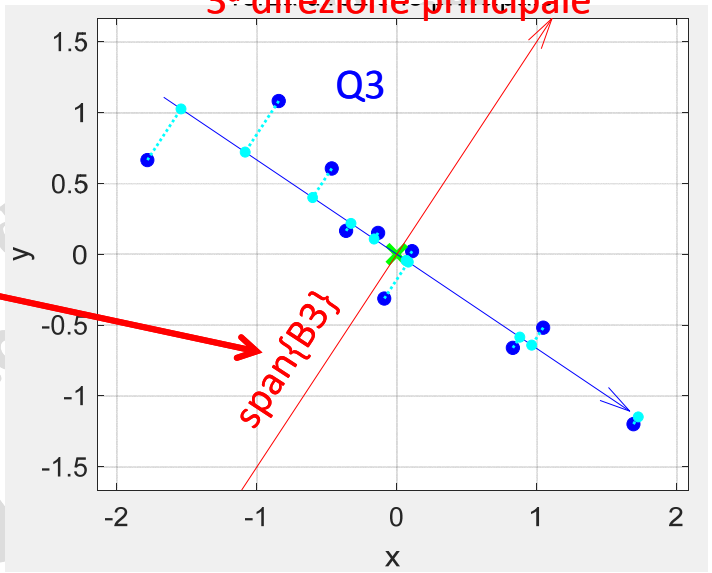
Illustrazione dell'algoritmo PCA incrementale (4)

complemento ortogonale di base2 sul piano $r^{\perp}$

```
B3=null(base2'); % base del complemento ortogonale di r
figure(2)
h=quiver(-2*B3(1),-2*B3(2),4*B3(1),4*B3(2),1);
set(h,'Color','r')
```



3ª direzione principale



proiezioni ortogonali su span{B3}

```
Pperp=base2*base2';  Pcmpl=eye(2) - Pperp;
Q5=Pcmpl*Q3'; Q5=Q5';
plot(Q5(:,1),Q5(:,2),'r.','MarkerSize',18);
[base3,comp3,lambda3]=pca(Q5,'NumComponents',1);
```

confronto con PCA completa

```
comp3 =
    -0.4339
  -0.061501
    0.14823
    0.43342
   0.078779
   0.051441
  -0.063018
  -0.090606
     0.2467
   -0.30955
```

```
comp =
    -0.85923     -1.8518     -0.4339
      1.9632      2.072    -0.061501
     -3.2115      1.158     0.14823
     0.28882     -1.3018     0.43342
    -0.30638    0.079027    0.078779
     -2.4193    -0.19491    0.051441
     -1.4579    -0.39336   -0.063018
    -0.16089      1.0563   -0.090606
      3.6831    -0.72228      0.2467
      2.4801    0.098876    -0.30955
```

```
[L3,w3]=powerMethod(Q5'*Q5/(N-1));
```

```
lambda3(1)
       0.064399
base3 =
       0.55422
       0.83237
```

```
L3 =
       0.064399
w3 =
       0.55422
       0.83237
```

Esempio PCA: interpretazione geometrica

```
N=10; X=mvnrnd([3 1 1],[1 .2 .7; .2 1 0; .7 0 1],N); % mvnrnd(μ, ∑, N)
```

base di $\mathbb{R}^3$

```
[basis,comp,lambda]=pca(X,'Economy',false);
```

v.c.

campioni

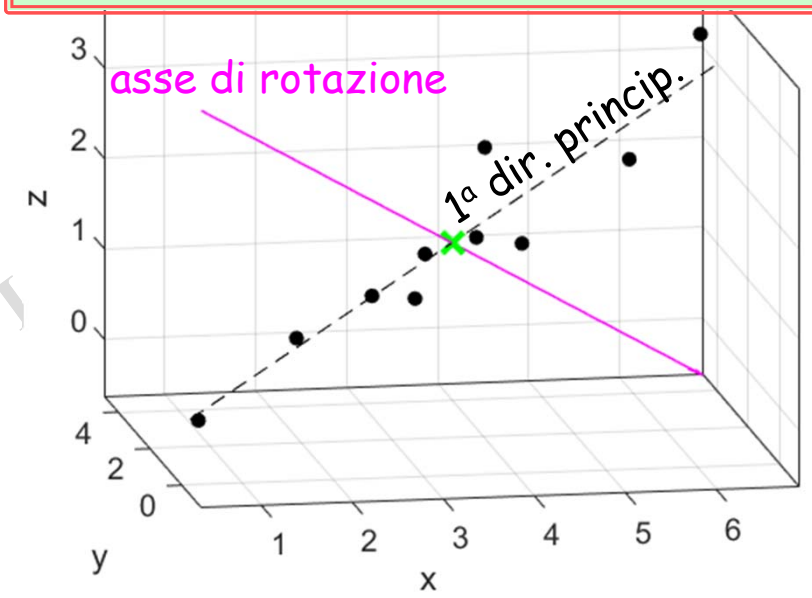
matrice dei dati

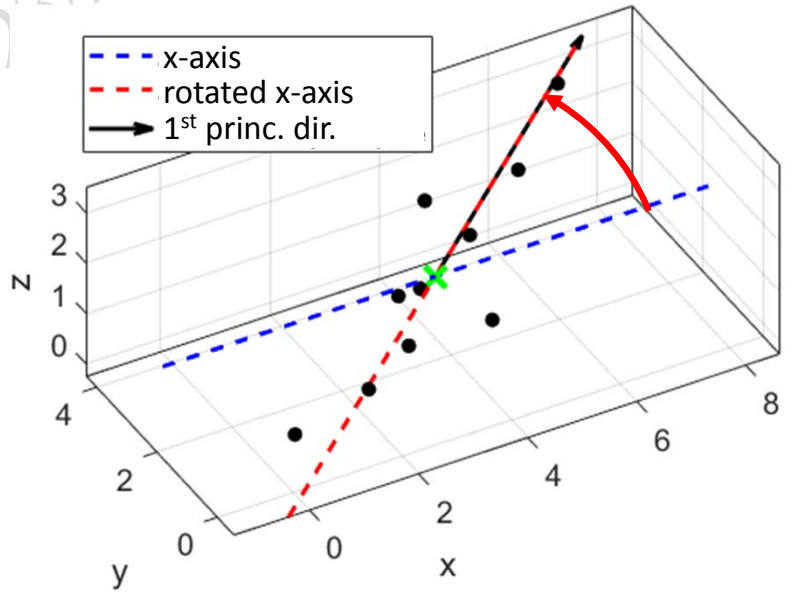
B=basis è una matrice ortogonale.

Se det(B)=1 ⇒ è una rotazione 3D attorno alla media campionaria

asse ed angolo della rotazione 3D

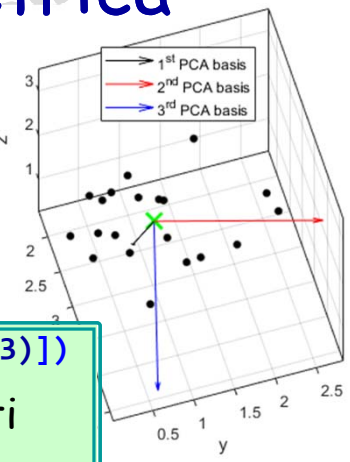
```
axis=null(basis - eye(size(basis,1)));
c=(trace(basis)-1)/2; % c: cos(θ), s: sin(θ)
s=(basis(2,1)-(1-c)*axis(1)*axis(2))/axis(3);
theta=atan2(s,c)*180/pi
theta = 49.38
```





Esempio PCA: interpretazione geometrica

```
N=20; mu=[3 1 2]; Sigma=[1 .2 .8; .2 1 0; .8 0 1];
X=mvnrnd(mu,Sigma,N); MU=mean(X);% media campionaria
[basis,comp,lambda]=pca(X);
fprintf("\ndet(PCA basis) = %g\n", det(basis))
det(PCA basis) = -1
```
è una rotazione impropria




```
E=eye(3); % base standard
disp([cross(E(:,1),E(:,2)) E(:,3)])
  0     0
  0     0
  1     1
```
vettori uguali
```
disp([cross(E(:,2),E(:,3)) E(:,1)])
  1     1
  0     0
  0     0
```
vettori uguali
```
disp([cross(E(:,3),E(:,1)) E(:,2)])
  0     0
  1     1
  0     0
```
vettori uguali

```
disp([cross(basis(:,1),basis(:,2)) basis(:,3)])
   -0.74938      0.74938
    0.20547     -0.20547
    0.62945     -0.62945
```
vettori opposti

Per ottenere una matrice di rotazione 3D, si può, ad es., cambiare segno ad una delle colonne di **basis***

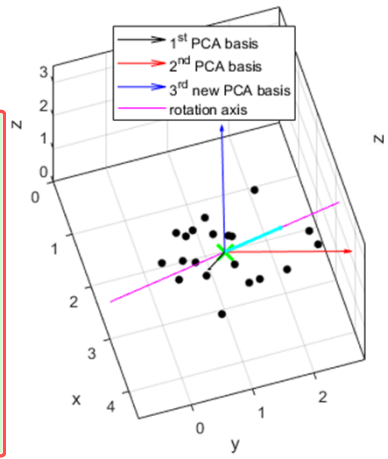
* Ovviamente, si dovrà cambiare segno anche alla corrispondente colonna di **comp**

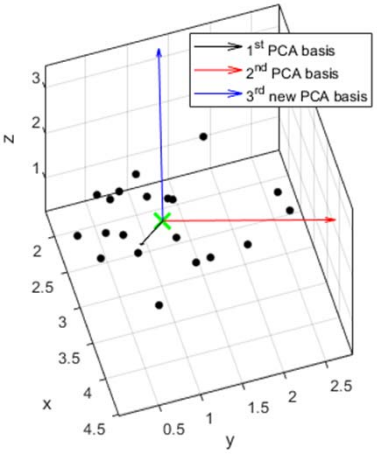
```
P=eye(3); P(3,3)=-1;
basis=basis*P; % nuova base (la 3ª colonna cambia segno)
fprintf("\ndet(PCA basis) = %g\n", det(basis))
det(PCA basis) = 1
```
ora è una rotazione propria

cross(basis(:,1),basis(:,2)) calcola una base per il complemento ortogonale di **basis(:,1:2)**

```
disp([cross(basis(:,1),basis(:,2)) null(basis(:,1:2)')])
   -0.74938     -0.74938
    0.20547      0.20547
    0.62945      0.62945
```
vettori uguali

```
ax=null(basis - eye(size(basis,1)))
ax =
      0.05614
      0.91699
      0.39494
```
asse ed angolo della rotazione 3D
```
c=(trace(basis)-1)/2; % c: cos(θ), s: sin(θ)
s=(basis(2,1)-(1-c)*ax(1)*ax(2))/ax(3);
theta=atan2(s,c)*180/pi
theta = -55.877
```





Derivazione dell'algoritmo per la PCA completa

Nello Spazio Lineare $\mathbb{R}^d$, avendo a disposizione d vettori linearmente indipendenti $\{\mathbf{u}_i\}$, questi possono costituire una sua base; pertanto $\forall \mathbf{x} \in \mathbb{R}^d$

$$\mathbf{U}=[\mathbf{u}_1,\ldots,\mathbf{u}_d]$$

$$\forall \mathbf{x} \in \mathbb{R}^d \Rightarrow \mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i = \mathbf{U}\mathbf{z}, \quad z_i = \mathbf{u}_i^{\mathsf{T}}\mathbf{x}, \quad \forall i = 1,\ldots,d$$

z_i: componente di $\mathbf{x}$ rispetto a $\mathbf{u}_i$

Ciò vale anche per gli N campioni del dataset $\mathcal{D} = \{\mathbf{x}^{[n]}\}_{n=1,\ldots,N}$.

Sia $\mathbf{x}'$ il vettore $\mathbf{x}$ con tutte le sue d componenti z_i espresse rispetto alla nuova base $\{\mathbf{u}_1,\ldots,\mathbf{u}_d\}$ e sia $\mathbf{x}''$ il vettore ottenuto considerandone solo le prime $K < d$ componenti:

$$\mathbf{x} = \mathbf{x}' = \sum_{i=1}^{d} z_i \mathbf{u}_i = \sum_{i=1}^{K} z_i \mathbf{u}_i + \sum_{i=K+1}^{d} z_i \mathbf{u}_i, \qquad \mathbf{x}'' = \sum_{i=1}^{K} z_i \mathbf{u}_i$$

$\mathbf{x}''$ è l'approssimazione di $\mathbf{x}'$ nel sottospazio generato dai primi K vettori $\mathbf{u}_i$

In $\mathbf{x}'$ si lascino liberi di variare gli ultimi $d-K$ scalari della combinazione lineare (b_i) per determinarli in seguito

$$\mathbf{U}_K = [\mathbf{u}_1,\ldots,\mathbf{u}_K] \qquad \widetilde{\mathbf{x}'} = \sum_{i=1}^{K} z_i \mathbf{u}_i + \sum_{i=K+1}^{d} b_i \mathbf{u}_i$$

Considerato l'intero dataset di N vettori $\mathcal{D}=\{\mathbf{x}^{[n]}\}_{n=1,\ldots,N}$, l'errore nel vettore $\mathbf{x}^{[n]}$ è dato da:

$$\mathbf{x}^{[n]} - \widetilde{\mathbf{x}^{[n]}} = \sum_{i=K+1}^{d} \left(z_i^{[n]} - b_i \right) \mathbf{u}_i$$

Si vogliono determinare i vettori della base $\mathbf{u}_i$ e i coefficienti b_i in modo da minimizzare la seguente funzione obiettivo che descrive l'errore nell'intero dataset:

$$\min_{\{\mathbf{u}_i\},\{b_i\}} J : J = \frac{1}{2} \sum_{n=1}^{N} \left\| \mathbf{x}^{[n]} - \widetilde{\mathbf{x}^{[n]}} \right\|_2^2 = \frac{1}{2} \sum_{n=1}^{N} \sum_{i=K+1}^{d} \left\| \left(z_i^{[n]} - b_i \right) \mathbf{u}_i \right\|_2^2 = \frac{1}{2} \sum_{n=1}^{N} \sum_{i=K+1}^{d} \left(z_i^{[n]} - b_i \right)^2$$

minimizzare SSE (sum of squared errors)

$$\min J : J = \frac{1}{2}\sum_{n=1}^{N}\left\|\mathbf{x}^{[n]} - \widetilde{\mathbf{x}^{[n]}}\right\|_2^2 = \frac{1}{2}\sum_{n=1}^{N}\sum_{i=K+1}^{d}\left\|\left(z_i^{[n]} - b_i\right)\mathbf{u}_i\right\|_2^2 = \frac{1}{2}\sum_{n=1}^{N}\sum_{i=K+1}^{d}\left(z_i^{[n]} - b_i\right)^2$$

Il funzionale J (somma di norme, cioè di funzioni convesse) avrà un minimo non banale ($\neq 0$) solo se si impongono dei vincoli. Per questi si può richiedere che i vettori della base $\{\mathbf{u}_i\}$ siano ortonormali: $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{ij}$, con δ_{ij} simbolo di Kronecker ($\delta_{ij}=1$ se $i=j$, $\delta_{ij}=0$ se $i\neq j$), vincoli riscritti come segue

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle - \delta_{ij} = \mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij} = 0.$$

Il problema di ottimizzazione vincolata con soli vincoli di uguaglianza può risolversi mediante il **metodo dei moltiplicatori di Lagrange**, risolvendo un problema di minimo libero per la seguente Funzione Lagrangiana (dove ogni vincolo, senza l'uguale a zero, è moltiplicato per uno scalare μ_{ij}):

$$\min_{\{b_i\},\{\mathbf{u}_i\},\{\mu_{ij}\}} \mathbb{L} : \mathbb{L} = \frac{1}{2}\sum_{n=1}^{N}\sum_{i=K+1}^{d}\left(z_i^{[n]} - b_i\right)^2 - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right)$$

moltiplicatori

Il problema di minimizzazione libera della funzione Lagrangiana $\mathbb{L}$ viene risolto annullando il suo gradiente (cioè cercando i punti stazionari di $\mathbb{L}$):

$$\frac{\partial \mathbb{L}}{\partial \mu_{hk}} = 0 \Rightarrow \frac{\partial \mathbb{L}}{\partial \mu_{hk}} = \sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\frac{\partial}{\partial \mu_{hk}}\left[\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right)\right] = \mathbf{u}_h^{\mathsf{T}}\mathbf{u}_k - \delta_{hk} = 0, \forall h,k \Rightarrow \{\mathbf{u}_i\} \text{ ortonormali come nei vincoli}$$

$$\frac{\partial \mathbb{L}}{\partial b_i} = 0 \Rightarrow \frac{\partial \mathbb{L}}{\partial b_i} = \frac{1}{2}\sum_{n=1}^{N}\sum_{j=K+1}^{d}\frac{\partial}{\partial b_i}\left[\cancel{\left(z_j^{[n]}\right)^2} + \left(b_j\right)^2 - 2z_j^{[n]}b_j\right] = \frac{1}{2}\sum_{n=1}^{N}\left[2b_i - 2z_i^{[n]}\right] = \boxed{Nb_i - \sum_{n=1}^{N}z_i^{[n]} = 0}$$

$$\frac{\partial \mathbb{L}}{\partial b_i} = 0 \Leftrightarrow b_i = \frac{1}{N}\sum_{n=1}^{N}z_i^{[n]} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_i^{\mathsf{T}}\mathbf{x}^{[n]} = \mathbf{u}_i^{\mathsf{T}}\bar{\mathbf{x}} \quad \text{dove} \quad \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{[n]} \text{ è la media campionaria}$$

Sostituendo b_i in $\mathbb{L}$, dopo aver invertito le due sommatorie, si ha:

$$\mathbb{L} = \frac{1}{2}\sum_{i=K+1}^{d}\sum_{n=1}^{N}\left(z_i^{[n]} - b_i\right)^2 - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right) =$$

$$= \frac{1}{2}\sum_{i=K+1}^{d}\sum_{n=1}^{N}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{x}^{[n]} - \mathbf{u}_i^{\mathsf{T}}\overline{\mathbf{x}}\right)^2 - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right) =$$

$$= \frac{1}{2}\sum_{i=K+1}^{d}\sum_{n=1}^{N}\left\{\mathbf{u}_i^{\mathsf{T}}\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\right\}^2 - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right) =$$

Per la propr. simmetrica del prodotto scalare std., il quadrato del prodotto scalare può scriversi come

$$\left\{\mathbf{u}_i^{\mathsf{T}}\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\right\}^2 = \left\langle \mathbf{u}_i, \left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\right\rangle\left\langle \mathbf{u}_i, \left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\right\rangle = \left\langle \mathbf{u}_i, \left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\right\rangle\left\langle \left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right), \mathbf{u}_i\right\rangle =$$

$$= \mathbf{u}_i^{\mathsf{T}}\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)^{\mathsf{T}}\mathbf{u}_i$$

prodotto esterno di 2 vettori

$$\mathbb{L} = \frac{1}{2}\sum_{i=K+1}^{d}\sum_{n=1}^{N}\left\{\mathbf{u}_i^{\mathsf{T}}\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\right\}^2 - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right) =$$

$\mathbf{u}_i$ non dipende da n

$$= \frac{1}{2}\sum_{i=K+1}^{d}\sum_{n=1}^{N}\left\{\mathbf{u}_i^{\mathsf{T}}\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)^{\mathsf{T}}\mathbf{u}_i\right\} - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right) =$$

$$= \frac{1}{2}\sum_{i=K+1}^{d}\mathbf{u}_i^{\mathsf{T}}\underbrace{\left\{\sum_{n=1}^{N}\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)\left(\mathbf{x}^{[n]} - \overline{\mathbf{x}}\right)^{\mathsf{T}}\right\}}_{\text{scatter matrix } \Sigma}\mathbf{u}_i - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right) =$$

(proporzionale alla matrice di covarianza campionaria)

$$\mathbb{L} = \frac{1}{2}\sum_{i=K+1}^{d}\mathbf{u}_i^{\mathsf{T}}\,\Sigma\,\mathbf{u}_i - \frac{1}{2}\sum_{i=K+1}^{d}\sum_{j=K+1}^{d}\mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right)$$

Resta da dimostrare che il minimo dell'errore J è raggiunto dagli autovalori/autovettori della scatter matrix (stessi autovettori della matrice di covarianza dei dati, autovalori scalati).

Per le proprietà 2 e 3 del gradiente del prodotto scalare std., risulta

$$\frac{\partial \mathbb{L}}{\partial \mathbf{u}_h} = \frac{1}{2}\frac{\partial}{\partial \mathbf{u}_h}\left[\sum_{i=K+1}^{d} \mathbf{u}_i^{\mathsf{T}}\, \Sigma\, \mathbf{u}_i\right] - \frac{1}{2}\frac{\partial}{\partial \mathbf{u}_h}\left[\sum_{i=K+1}^{d}\sum_{j=K+1}^{d} \mu_{ij}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_j - \delta_{ij}\right)\right] =$$

$$= \frac{1}{2}\frac{\partial}{\partial \mathbf{u}_h}\left[\mathbf{u}_h^{\mathsf{T}}\, \Sigma\, \mathbf{u}_h\right] - \frac{1}{2}\frac{\partial}{\partial \mathbf{u}_h}\left[\mu_{hh}\mathbf{u}_h^{\mathsf{T}}\mathbf{u}_h\right] =$$

Propr. 3

$$= \frac{1}{2}\left[2\, \Sigma\, \mathbf{u}_h\right] - \frac{1}{2}\mu_{hh}\frac{\partial}{\partial \mathbf{u}_h}\left[\mathbf{u}_h^{\mathsf{T}}\mathbf{u}_h\right] = \Sigma\, \mathbf{u}_h - \frac{1}{2}\mu_{hh}\left[2\mathbf{u}_h\right] =$$

Propr. 2

scatter matrix

autovalori

autovettori

$$\Sigma\, \mathbf{u}_h - \mu_{hh}\mathbf{u}_h = 0$$

tipica equazione matriciale per autovalori/autovettori

⇨

Inoltre, l'errore totale (SSE) è dato dalla somma dei rimanenti $d-K$ autovalori (con un fattore 1/2).

$$J = \frac{1}{2}\sum_{n=1}^{N}\left\|\mathbf{x}^{[n]} - \widetilde{\mathbf{x}^{[n]}}\right\|_2^2 = \frac{1}{2}\sum_{i=K+1}^{d} \mathbf{u}_i^{\mathsf{T}}\, \Sigma\, \mathbf{u}_i = \frac{1}{2}\sum_{i=K+1}^{d} \mathbf{u}_i^{\mathsf{T}}\lambda_i\mathbf{u}_i = \frac{1}{2}\sum_{i=K+1}^{d} \lambda_i\mathbf{u}_i^{\mathsf{T}}\mathbf{u}_i = \frac{1}{2}\sum_{i=K+1}^{d} \lambda_i$$

Algoritmo PCA completa

Se si vogliono **tutte** le componenti principali, l'*algoritmo* PCA *incrementale* **non è conveniente** computazionalmente.

In tal caso:

- calcolare efficientemente ed accuratamente tutti gli autovalori ed autovettori della matrice di covarianza campionaria (o della scatter matrix), oppure alternativamente della matrice di correlazione.
- ordinare gli autovalori in ordine decrescente (e riordinare di conseguenza i corrispondenti autovettori).

Per la riduzione della dimensionalità dello spazio dei dati:

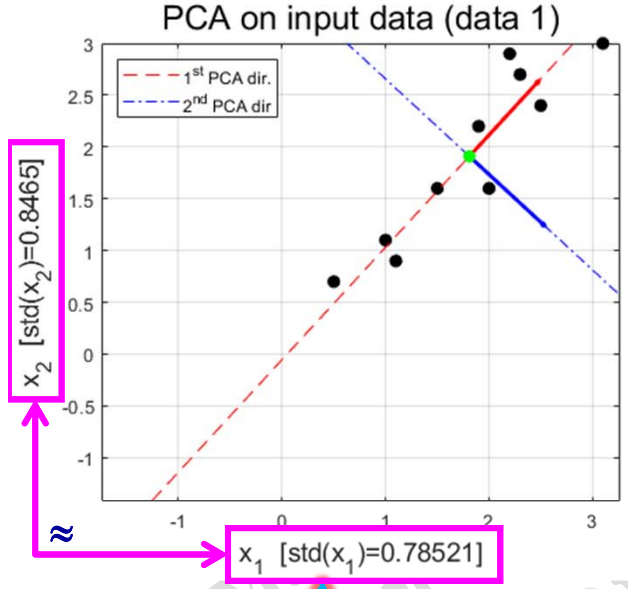
- calcolare la somma cumulativa normalizzata degli autovalori.
- calcolare il minimo indice K tale che la somma cumulativa normalizzata sia maggiore o uguale ad una tolleranza percentuale.
- selezionare i primi K autovettori come base del sottospazio PCA "ridotto".
- calcolare le componenti dei dati risp. a tale base PCA "ridotta".

Matrice di covarianza, scatter matrix, o di correlazione:

Esempio 1

```
x1=[2.5 0.5 2.2 1.9 3.1 2.3 2.0 1.0 1.5 1.1];
x2=[2.4 0.7 2.9 2.2 3.0 2.7 1.6 1.1 1.6 0.9];
X=[x1;x2]; mu=mean(X,2); Xc=X - mu; Xs=normalize(X,2);
```

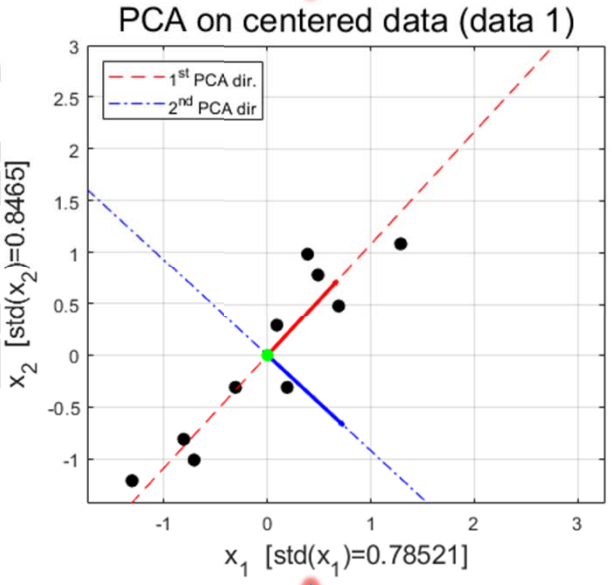
direzioni PCA quasi uguali



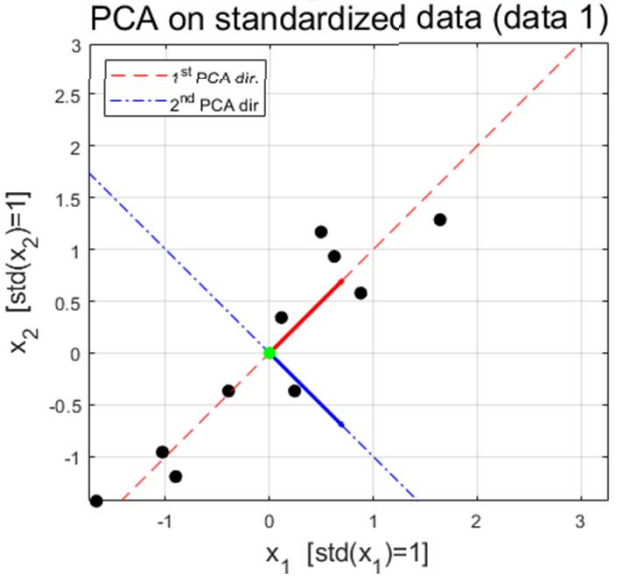
≈

campioni con deviazione standard simile

dati scalati e direzioni PCA diverse solo per un angolo ~2°




```
basis =
   0.70711    0.70711
   0.70711   -0.70711
```



campioni e base PCA sono traslati da μ in $\mathbf{O}$

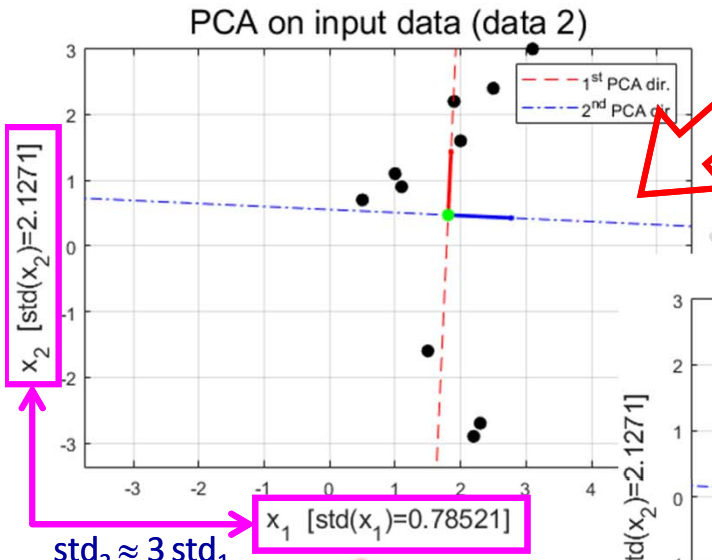
Matrice di covarianza, scatter matrix, o di correlazione:

Esempio 2

x2: max varianza

```
x1=[2.5 0.5  2.2 1.9 3.1  2.3 2.0 1.0  1.5 1.1];
x2=[2.4 0.7 -2.9 2.2 3.0 -2.7 1.6 1.1 -1.6 0.9];
X=[x1;x2]; mu=mean(X,2); Xc=X - mu; Xs=normalize(X,2);
```

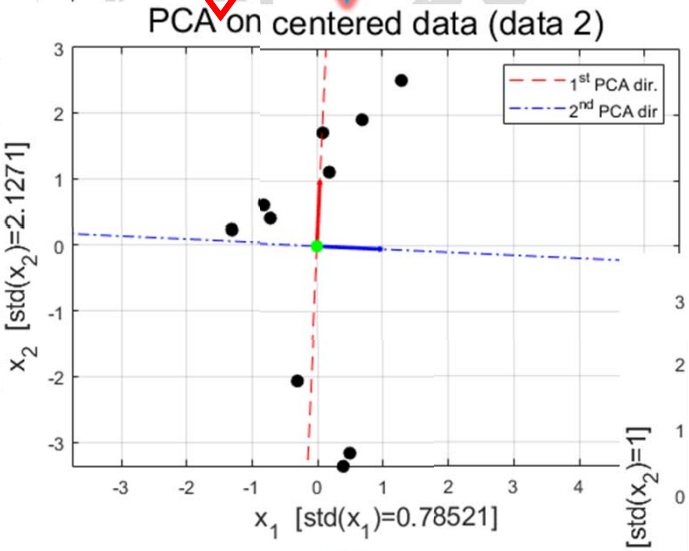
1ª direzione PCA quasi verticale



$std_2 \approx 3\, std_1$

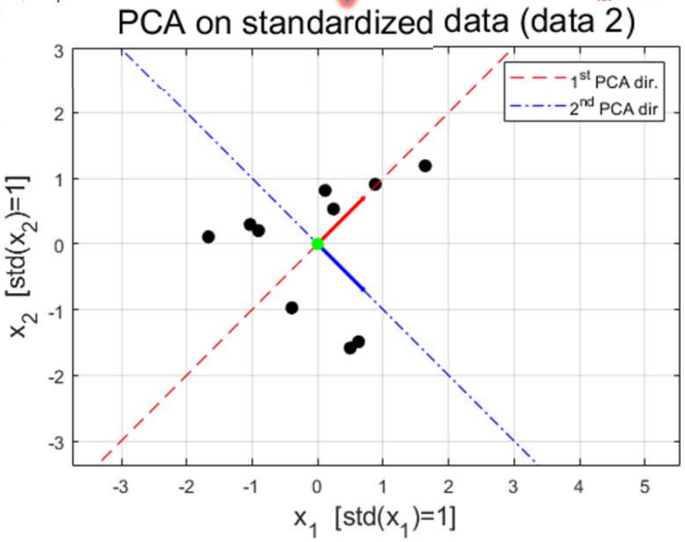
campioni con deviazione standard diversa

dati scalati e direzioni PCA differenti



stessa base PCA dell'es. 1

```
basis =
   0.70711    0.70711
   0.70711   -0.70711
```



campioni e base PCA sono traslati da μ in $\mathbf{O}$

la PCA sui dati favorisce la v.c. con massima varianza

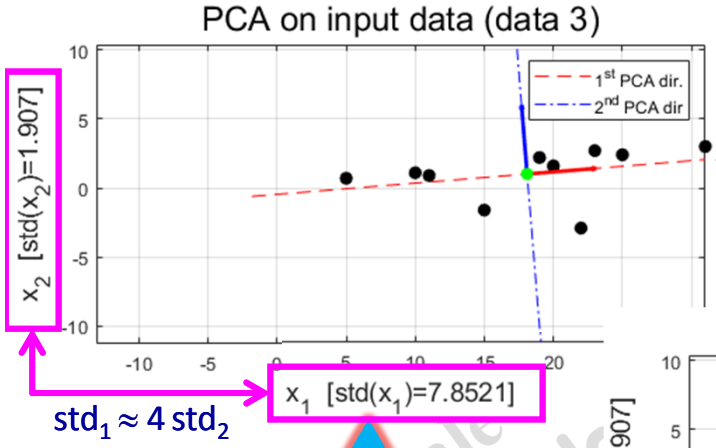
Matrice di covarianza, scatter matrix, o di correlazione:

Esempio 3

10×**x1** precedente

```
x1=[25   5   22   19   31   23   20   10   15   11];
x2=[ 2.4 0.7 -2.9  2.2  3.0  2.7  1.6  1.1 -1.6  0.9];
X=[x1;x2]; mu=mean(X,2); Xc=X - mu; Xs=normalize(X,2);
```

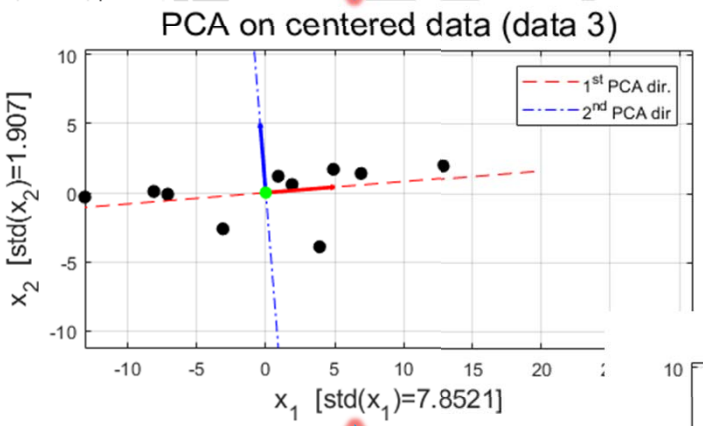
1ª direzione PCA quasi orizzontale



$std_1 \approx 4\, std_2$

campioni con deviazione standard diversa

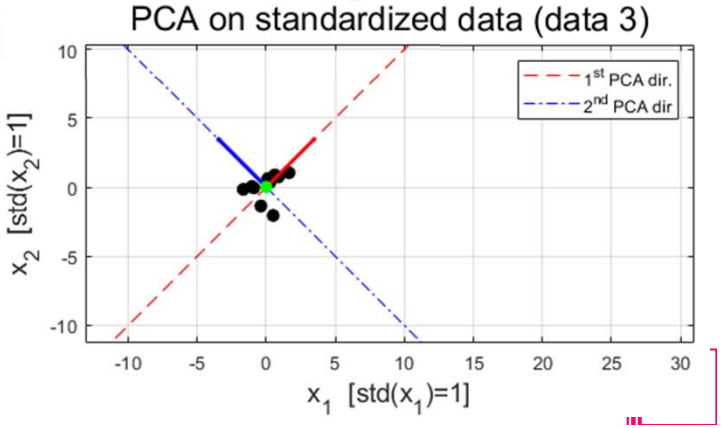
dati scalati e direzioni PCA differenti



stessa base PCA dell'es. 1

cambia solo il verso del 2° vettore

```
basis =
   0.70711   -0.70711
   0.70711    0.70711
```



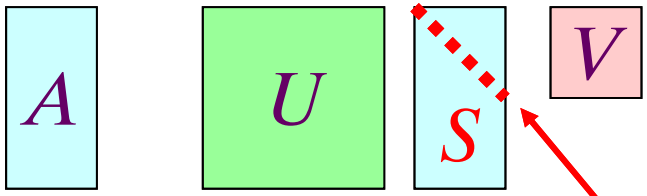
campioni e base PCA sono traslati da μ in $\mathbf{O}$

la PCA sui dati favorisce la v.c. con massima varianza

Calcolare autovalori/autovettori con SVD

Richiami: "Singular Value Decomposition" di $A_{m\times n}$

$$A_{m\times n} = U_{m\times m}\cdot S_{m\times n}\cdot V^{\mathsf{T}}{}_{n\times n}$$



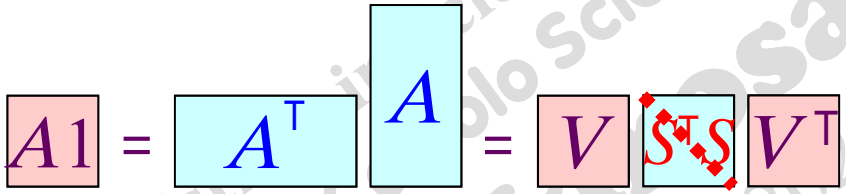
valori singolari σ_j di A

colonne ortonormali

$$U^{\mathsf{T}}\cdot U = I_{m\times m}$$

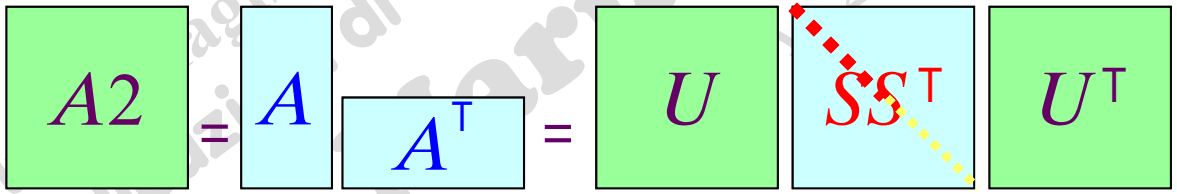
$$V^{\mathsf{T}}\cdot V = I_{n\times n}$$

$$A1 = A^{\mathsf{T}}\cdot A = V\cdot S^{\mathsf{T}}\cdot U^{\mathsf{T}}\cdot U\cdot S\cdot V^{\mathsf{T}} = V\cdot\left(S^{\mathsf{T}}\cdot S\right)\cdot V^{\mathsf{T}}$$



autovettori

autovalori σ_j^2



$$A2 = A\cdot A^{\mathsf{T}} = U\cdot S\cdot V^{\mathsf{T}}\cdot V\cdot S^{\mathsf{T}}\cdot U^{\mathsf{T}} = U\cdot\left(S\cdot S^{\mathsf{T}}\right)\cdot U^{\mathsf{T}}$$

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

ACS parte 2: ACS_09c

Argomenti trattati

- **Applicazione della PCA: l'algoritmo "Eigenfaces".**
- **PCA versus Least Squares.**

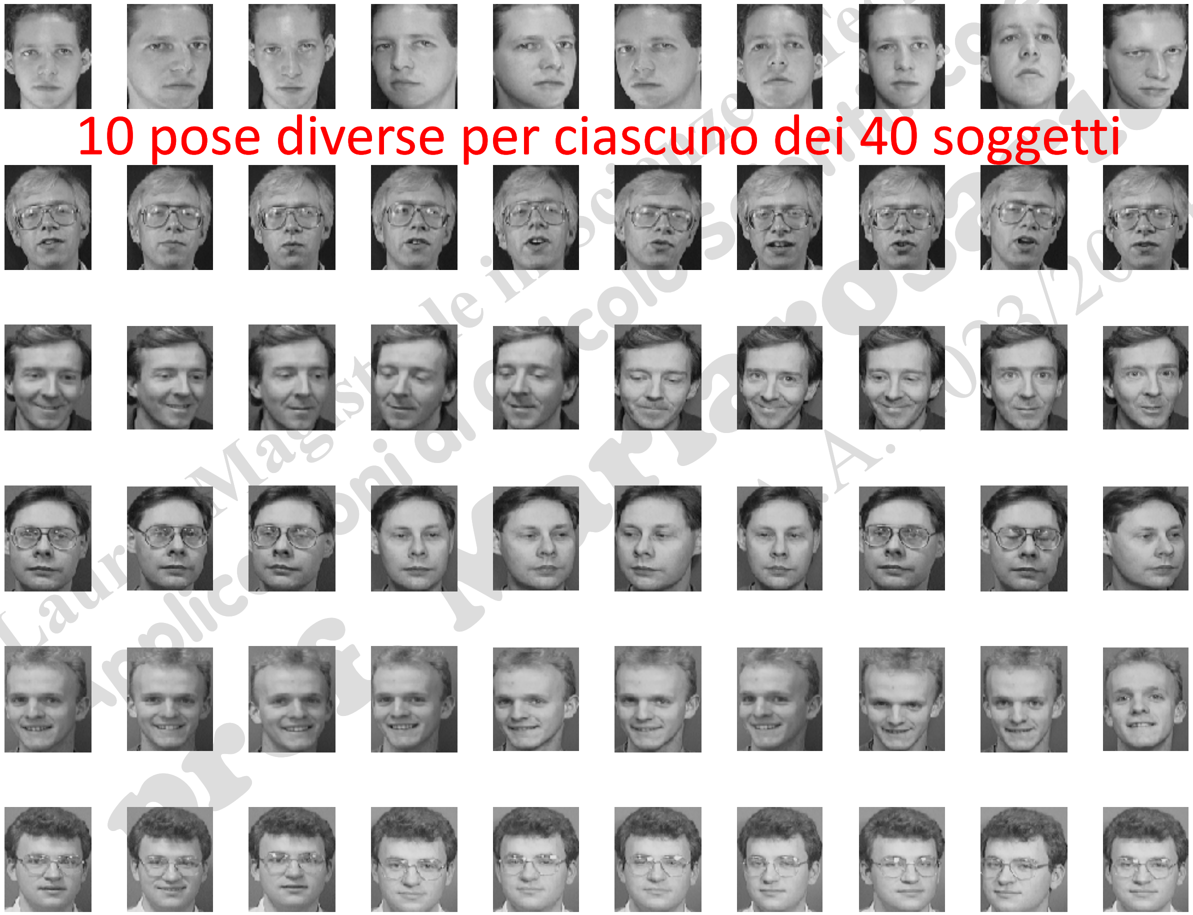
Algoritmo “Eigenfaces” (PCA)

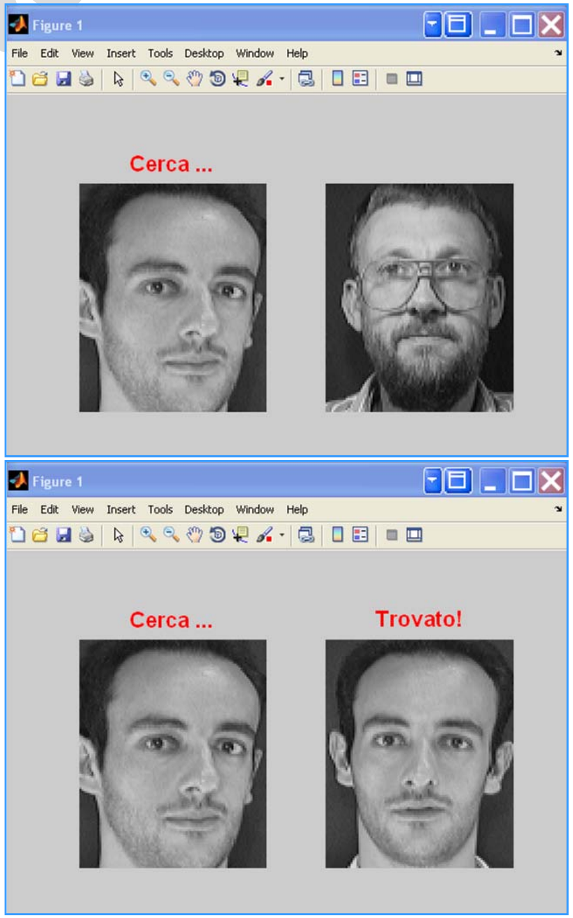
Algoritmo “Autofacce”: riconoscere un soggetto tra molti

database di facce* = 400 images (112×92 pxls) = **4 121 600 bytes**

* Download il mat-file: db_400faces_112x92_col_uint8.mat per il database di facce

- ❑ PCA: riduce lo spazio per memorizzare i dati (compressione di dati).
- ❑ PCA: rende più veloce cercare una faccia nel database.





Algoritmo Eigenfaces: Idea

Il problema è trattato da un punto di vista statistico (PCA): ordinare variabili random per varianze decrescenti.

È possibile estrarre, dalla popolazione di tutte le facce, una "base PCA ridotta" in grado di mantenere l'informazione sui dati entro una tolleranza prestabilita?

Compressione di immagini: invece di memorizzare le immagini, si memorizzano solo i loro "*descrittori*" (...), il cui numero è molto minore del numero totale dei pixel delle immagini.

Ricerca facciale più veloce: invece di confrontare i pixel delle immagini (molto lento e inutile!), si confrontano solo i loro "descrittori" (...) , il cui numero è molto minore del numero totale dei pixel delle immagini. Inoltre è più generale che cercare una corrispondenza esatta tra pixel (stessa immagine).

Algoritmo Eigenfaces: com'è organizzato il database

La k^sima immagine (112×92 pxls) è salvata in un vettore (colonna) X_k, di 10304=112×92 componenti, e tutte queste colonne formano una matrice **W** di **uint8*** (interi senza segno a 8-bit).

* Download il mat-file: db_400faces_112x92_col_uint8.mat per il database delle facce

```
load db_400faces_112x92_col_uint8 % mat-file del database
whos
Name          Size                Bytes    Class
  W         10304x400             4121600  uint8
```

```
m=112; n=92;
imshow(reshape(W(:,346),m,n))
```

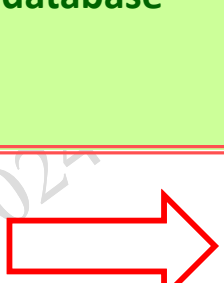


immagine #346

X(10304 ×400) è una matrice di numeri double, per i calcoli floating-point

```
X = double(W);
```

per visualizzare le immagini

```
imshow(uint8(reshape(X(:,346),m,n)))
```

```
image(uint8(reshape(X(:,346),m,n)))
```

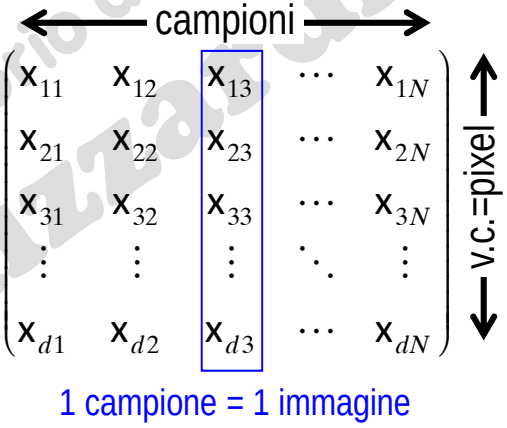
per visualizzare una matrice di numeri come immagine

```
imagesc(reshape(X(:,346),m,n))
axis equal; axis off
colormap('gray')
```

Esempio MATLAB: Algoritmo "Eigenfaces"

```
load db_400faces_112x92_col_uint8
```

W(10304,400): le immagini sono uint8



1 campione = 1 immagine

1) Carica nel Workspace di MATLAB il database di facce numerico: la matrice dei dati **W**($d \times N$) contiene un campione in ogni colonna. Lo *spazio delle v.c.* ha d dimensioni. Il database completo di facce contiene N=400 immagini di size 112×92 pxls. La dimensione dello *spazio delle v.c.* è: d=10304.

Soggetto cercato

#380

```
ri = 380;
```

costante oppure random

```
ri = randi(size(X,2));
```

```
X = double(W(:,[1:ri-1 ri+1:end]));
```

double: per calcoli float-point

2) Sceglie l'indice **ri** della faccia da cercare, nel database completo, e rimuove il vettore corrispondente dalla matrice dei dati (l'immagine cercata non è esattamente nel database). L'indice **ri** può essere scelto costante o random.

```
Nc = size(X,2);
mu = mean(X,2);
Xc = X - repmat(mu,1,Nc);
```

```
w=uint8(reshape(mu,m,n));
imshow(w)
xlabel('mean face')
```

faccia media

3) Calcola il vettore μ (media campionaria) e la matrice centrata dei dati $\mathbf{X}_c$.

Se la faccia da cercare è #380, nel database, dello stesso soggetto sono presenti altre 9 pose diverse


#371


#372


#373


#374


#375


#376


#377


#378


#379


#380

4) Applica la PCA:

```
[basis,comp,lambda] = pca(Xc','Economy',false);
```

stessi risultati di

```
[basis,comp,lambda] = pca(X','Economy',false);
```

```
[basis,comp,lambda] = pca(Xc','Economy',true);
disp(numel(lambda))
    398
```

PCA *cambia base* allo spazio dei *vettori dei dati centrati* e produce una nuova base di $\mathbb{R}^d$, B=**basis'**, *ortonormale* (cioè $BB^T = B^TB = I$):

basis' — perché pca vuole **Xc'** e non **Xc**

proiezione **PCA** : $\mathbf{v} \in X_C \rightarrow \mathbf{w} = B^T\mathbf{v} \in$ *sottospazio* PCA

ricostruzione **PCA**$^{-1}$: $\mathbf{w} \in$ *sottospazio* PCA $\rightarrow \mathbf{v} = B\,\mathbf{w} \in X_C$

w = comp'

Non dimenticare che si sta lavorando con vettori di uno Spazio Lineare

proiezione di X_C sul sottospazio PCA

```
w=basis'*Xc;
disp(size(w))
       10304         399
all(all(abs(comp'-w)<1e-9))
ans =
  logical
   1
```

comp' : componenti risp. alla base PCA

base ortonormale di $\mathbb{R}^{10304}$

```
disp(size(basis))
       10304       10304
all(all(abs(round(basis*basis')-eye(10304))<1e-9))
ans =
  logical
   1
all(all(abs(round(basis'*basis)-eye(10304))<1e-9))
ans =
  logical
   1
```

ricostruzione X_C dal sottospazio PCA

```
v=basis*comp'; % *w
disp(size(v))
       10304         399
all(all(abs(Xc-v)<1e-9))
ans =
  logical
   1
```

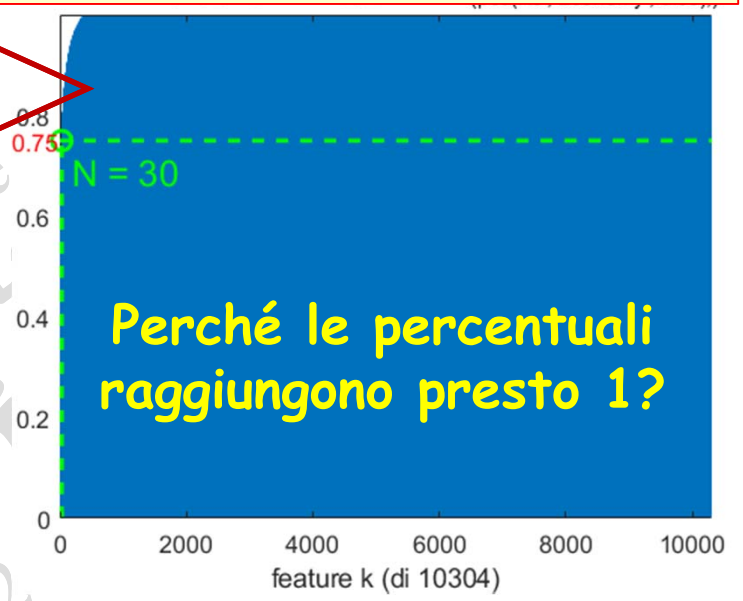
La **PCA** può essere calcolata numericamente mediante gli *autovalori* ed *autovettori* della matrice di covarianza oppure mediante fattorizzazione SVD di X_c.

elapsed time di **eig** di cov: 89.72 sec
elapsed time di **svd** di X_c : 4.767 sec
elapsed time di **pca** di X : 3.912 sec

5) Per stabilire un opportuno size per la base PCA ridotta, calcola la somma cumulativa normalizzata della "varianza spiegata" ("explained variance") delle *v.c.* (*features*):

```
PERCtol=0.75; % percentuale di tolleranza
PERC=cumsum(lambda)/sum(lambda);
N=find(PERC >= PERCtol,1,'first');
Rbasis=basis(:,1:N);
```

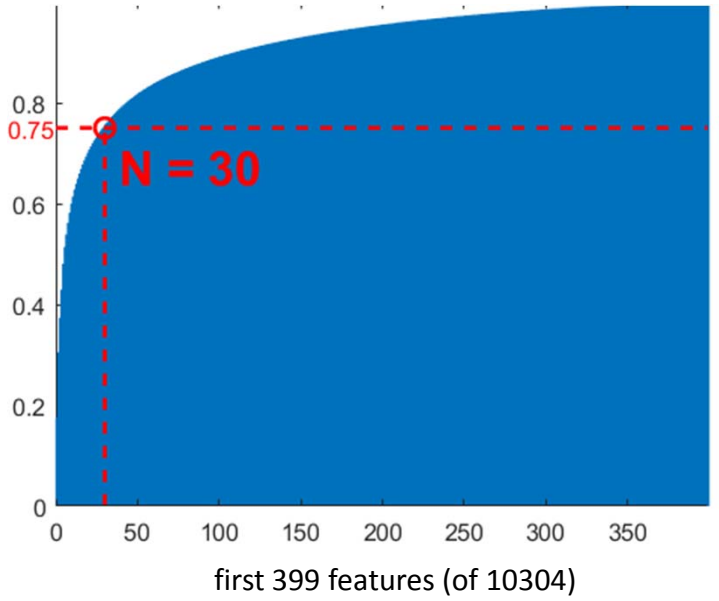
base PCA ridotta



L'algoritmo ha ridotto la base dello spazio dei pixel da 10304 a 30 vettori.

Le prime 8 Autofacce della … base "spettrale" (base PCA)

"spettrale" come da autovalori, … ma anche come fantasmi



proiezione sul sottospazio PCA

```
Rcomp = Rbasis'*Xc;
disp(size(Rcomp))
    30   399
disp(size(comp'))
 10304   399
```

⟺

```
Rcomp1 = comp(:,1:N)';
all(all(abs(Rcomp1-Rcomp) < 1e-9))
ans =
  logical
   1
```

6) Calcola il database ridotto, cioè tutte le componenti delle facce rispetto alla base PCA ridotta.

Il database delle facce, rispetto alla base standard, ha dimensioni **10304×399**.

Ora, rispetto alla base PCA ridotta, le sue dimensioni sono **30×399** (compressione dei dati).

```
Wb=Rbasis'*(double(W(:,ri)) - mu);
```

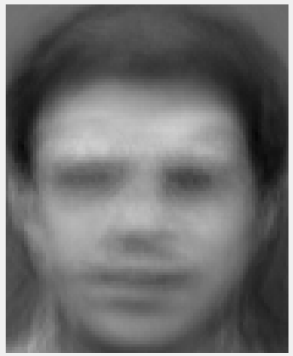
immagine da cercare

matematicamente equivalente a

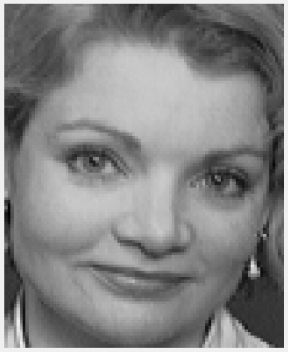
```
Wr=Rbasis\(double(W(:,ri)) - mu);
all((abs(Wb-Wr(1:N)) < 1e-9))
ans =
  logical
   1
```

7) Anche l'immagine da cercare è proiettata sul sottospazio PCA.

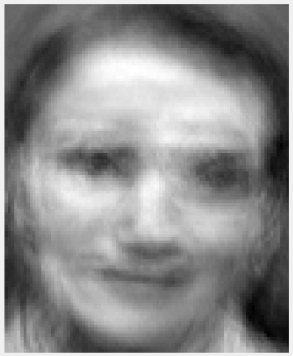
faccia originale cercata



ricostruzione della faccia cercata



faccia #346



ricostruzione della faccia #346

8) Cerca la faccia voluta nel nuovo database ridotto.

Per trovare il soggetto usa due metriche diverse (solo per confrontarne i risultati)

```
DIST = zeros(Nc,1);     SIML = zeros(Nc,1);
for j=1:Nc
    % metrica di distanza: norma euclidea del vettore residuo
    DIST(j) = norm(Rcomp(:,j) - Wb);
    % metrica di similarità: coseno dell'angolo tra i vettori
    SIML(j) = dot(Rcomp(:,j),Wb)/(norm(Rcomp(:,j))*norm(Wb));
end
```

$$\|\vec{u}-\vec{v}\|_2$$

$$\cos\theta = \frac{\langle \vec{u},\vec{v} \rangle}{\|\vec{u}\|_2 \|\vec{v}\|_2}$$

forma scalare

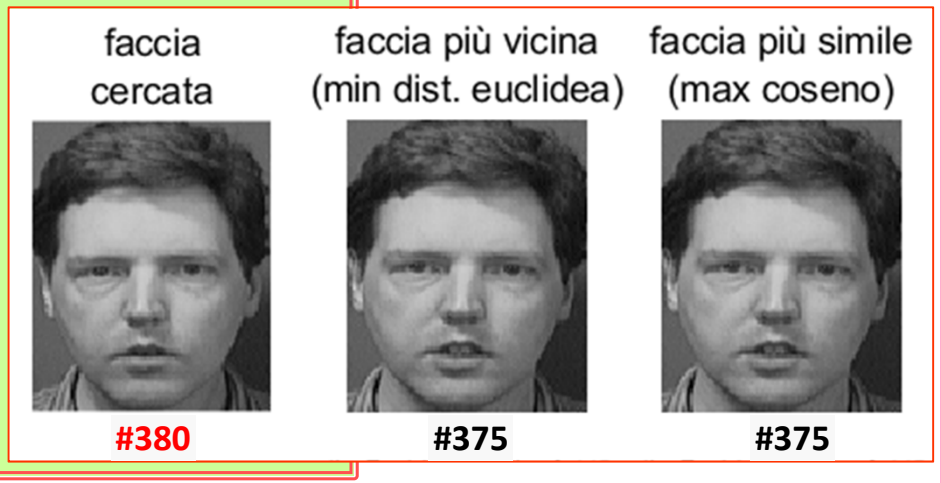
```
DIST=vecnorm(Rcomp-Wb); DIST=DIST';
SIML=(Rcomp'*Wb)/norm(Wb)./(vecnorm(Rcomp))';
```

forma vettoriale

```
% i migliori risultati
Jdist = find(DIST == min(DIST),1,'first');
Jsiml = find(SIML == max(SIML),1,'first');
```

```
% corregge l'indice dell'immagine (per riportarlo al database intero)
if Jdist >= ri
    JJdist=Jdist+1;
else
    JJdist=Jdist;
end
if Jsiml >= ri
    JJsiml=Jsiml+1;
else
    JJsiml=Jsiml;
end
```

solo per visualizzare



9) Cerca i 4 migliori risultati:

```
PERCtol=0.75; % tolleranza percentuale
```

```
[DIST,J1] = sort(DIST); % euclidean norm
for k=1:4
    if J1(k) >= ri
        JJ=J1(k)+1;
    else
        JJ=J1(k);
    end
    JJ
end
```

```
[SIML,J2] = sort(SIML,'descend'); % cosine
for k=1:4
    if J2(k) >= ri
        JJ=J2(k)+1;
    else
        JJ=J2(k);
    end
    JJ
end
```

corregge l'indice delle immagini per riportarle all'intero database

soggetto cercato #380

1ª faccia più vicina #375

dist/max(dist)=0.11523
coseno = 0.95971

2ª faccia più vicina #372

dist/max(dist)=0.14928
coseno = 0.93431

3ª faccia più vicina #377

dist/max(dist)=0.20234
coseno = 0.86906

4ª faccia più vicina #373

dist/max(dist)=0.24896
coseno = 0.82538

1ª faccia più simile #375

dist/max(dist)=0.11523
coseno = 0.95971

2ª faccia più simile #372

dist/max(dist)=0.14928
coseno = 0.93431

3ª faccia più simile #377

dist/max(dist)=0.20234
coseno = 0.86906

4ª faccia più simile #376

dist/max(dist)=0.27
coseno = 0.82578

PERCtol=0.75; % tolleranza percentuale

un'altra ricerca

#157

metrica: $\cos\theta = \dfrac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\|_2 \, \|\vec{v}\|_2}$

1ª faccia più vicina #156

dist/max(dist)=0.65734
coseno = 0.055058

2ª faccia più vicina #152

dist/max(dist)=0.655
coseno = 0.063914

3ª faccia più vicina #153

dist/max(dist)=0.65608
coseno = 0.060561

4ª faccia più vicina #146

dist/max(dist)=0.65049
coseno = 0.10322

metrica: $\|\vec{u} - \vec{v}\|_2$

1ª faccia più simile #156

dist/max(dist)=0.65734
coseno = 0.81892

2ª faccia più simile #152

dist/max(dist)=0.655
coseno = 0.65951

3ª faccia più simile #155

dist/max(dist)=0.65714
coseno = 0.58542

4ª faccia più simile #153

dist/max(dist)=0.65608
coseno = 0.56828

Esercizio

Implementare l'algoritmo* "Eigenfaces", e calcolare gli autovalori/autovettori mediante le funzioni MATLAB:

`pca()`: PCA of X, e di Xc, e di Xs;

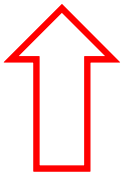
`svd()`: fattorizzazione SVD di Xc, e di Xs;

`eig()`: autovalori/autovettori di Xc*Xc', e di Xs*Xs';

confrontandone i risultati.

Confrontare anche i loro tempi di esecuzione con

`tic`, …, **`T=toc`**.



inserire qui il codice da valutare

* Download il mat-file db_400faces_112x92_col_uint8.mat per il database di facce

Vedere l'Help Browser di MATLAB per usare **`tic`**, …, **`T=toc`**.

Esercizio

Scrivere una funzione MATLAB per implementare l'algoritmo "PCA incremetale" (dove il massimo autovalore ed il corrispondente autovettore sono calcolati dalla funzione "powerMethod"), confrontandone i risultati con quelli restituiti dalla funzione MATLAB **`pca()`**.

Confrontare anche i tempi di esecuzione mediante

`tic`, …, **`T=toc`**.

Retta di regressione in $\mathbb{R}^2$

In Statistica la retta dei Minimi Quadrati Lineari ("Linear Least Squares") è detta **retta di regressione**.

A differenza della **correlazione**, che indica una dipendenza stocastica tra due variabili casuali x and y, la **regressione** denota una dipendenza funzionale tra x e y:

- $y=\Phi(x)$, cioè y dipende da x,
- o $x=\Psi(y)$, cioè x dipende da y.

In particolare, per la **regressione lineare** $\Phi(x)$, o $\Psi(y)$, è una funzione lineare.

- Regressione Lineare di y risp. a x: $y=\Phi(x)=ax+b$

  Si deve minimizzare il seguente funzionale

$$J_{LS}^{(1)}(a,b)=\sum_{i=1}^{n}\left[y_i-(ax_i+b)\right]^2$$

- Regressione Lineare di x risp. a y: $x=\Psi(y)=cy+d$

  Si deve minimizzare il seguente funzionale

$$J_{LS}^{(2)}(c,d)=\sum_{i=1}^{n}\left[x_i-(cy_i+d)\right]^2$$

Retta di regressione di y risp. a x in $\mathbb{R}^2$

Per definizione la retta "Least Squares" r : $y=ax+b$, relativa ai dati $(x_i,y_i)_{i=1,\dots,n}$, minimizza la somma dei quadrati dei residui: min $J_{LS}(a,b)$

$$J_{LS}^{(1)}(a,b)=\sum_{i=1}^{n}\underbrace{\left[y_i-(ax_i+b)\right]}_{\text{residui}}^2$$

$$\nabla J_{LS}(a,b)=0 \quad \begin{cases}\dfrac{\partial}{\partial a}J_{LS}^{(1)}=0 \\ \dfrac{\partial}{\partial b}J_{LS}^{(1)}=0\end{cases}$$

$$\frac{\partial}{\partial a}J_{LS}^{(1)}=0 \Leftrightarrow 2\sum_{i=1}^{n}\left[y_i-(ax_i+b)\right]x_i=0$$

$$\frac{\partial}{\partial b}J_{LS}^{(1)}=0 \Leftrightarrow 2\sum_{i=1}^{n}\left[y_i-(ax_i+b)\right]=0$$

$$J_{LS}^{(1)}(a,b)=\left\|\begin{pmatrix}x_1 & 1\\ x_2 & 1\\ \vdots & \vdots\\ x_n & 1\end{pmatrix}\begin{pmatrix}a\\ b\end{pmatrix}-\begin{pmatrix}y_1\\ y_2\\ \vdots\\ y_n\end{pmatrix}\right\|_2^2=\left\|A\begin{pmatrix}a\\ b\end{pmatrix}-\mathbf{y}\right\|_2^2$$

$$\frac{1}{n}\sum_{i=1}^{n}\left[y_i-(ax_i+b)\right]=0$$

media

$$\frac{1}{n}\sum_{i=1}^{n}y_i=a\frac{1}{n}\sum_{i=1}^{n}x_i+\frac{1}{n}\sum_{i=1}^{n}b$$

$$\frac{\partial}{\partial b}J_{LS}^{(1)}=0 \Rightarrow \boxed{\bar{y}=a\bar{x}+b}$$

cioè la retta Least Squares, per i dati $(x_i,y_i)_{i=1,\dots,n}$, passa per il punto:

media campionaria $P_\mu\equiv(\bar{x},\bar{y})=\left(\frac{1}{n}\sum_{i=1}^{n}x_i,\frac{1}{n}\sum_{i=1}^{n}y_i\right)$

Retta Least Squares in $\mathbb{R}^2$ con MATLAB

$$r : y = ax + b$$

```
%% P: 2D punti campione da 𝒩(μ,Σ)
N=10;  P=mvnrnd([3 1],[1 .2; .2 .7],N);
B0=mean(P); % media campion. (baricentro)
```

1° metodo: solo per $\mathbb{R}^2$

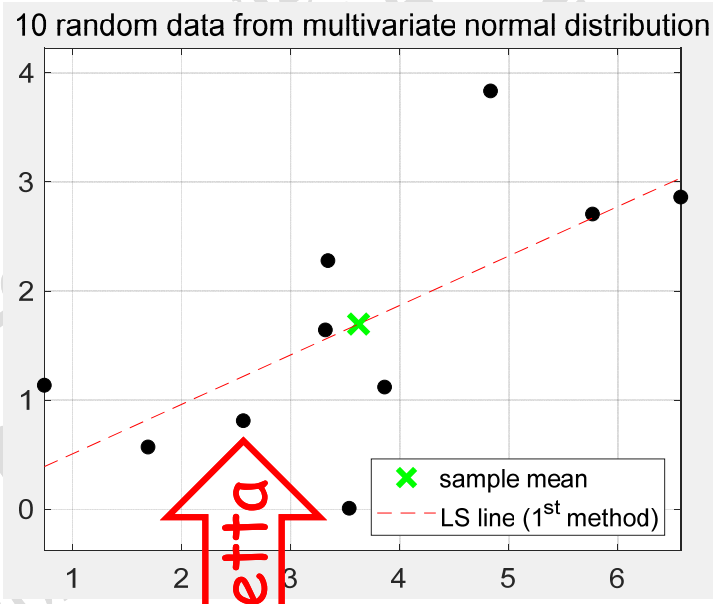
```
%% retta Least Squares - 1° metodo
Xi=P(:,1); Yi=P(:,2); X=[min(Xi) max(Xi)];
coef1=polyfit(Xi,Yi,1);
Y=polyval(coef1,X);
```

2° metodo: sistema sovradeterminato più *generale*

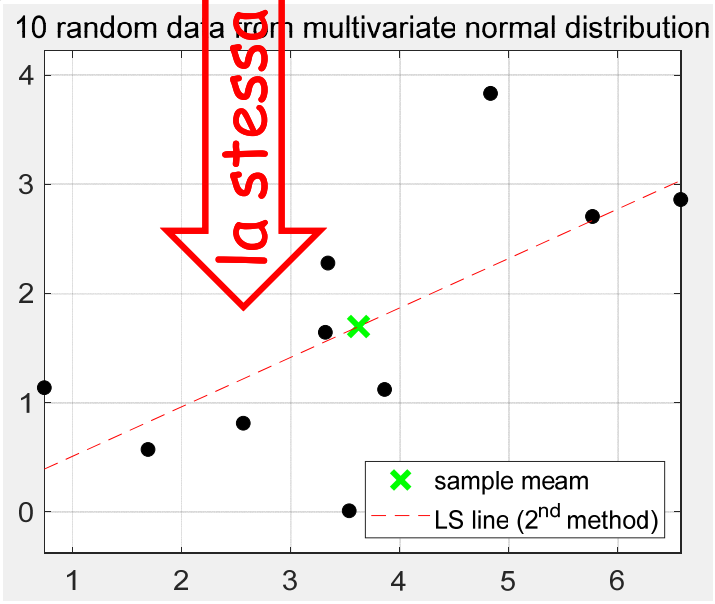
$$y_i = ax_i + b, \quad i = 1,\ldots,n$$

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

```
%% retta Least Squares - 2° metodo
Xi=P(:,1); Yi=P(:,2); X=[min(Xi) max(Xi)];
A=[Xi ones(N,1)];  coef2=A\Yi;
Y=polyval(coef2,X);
```



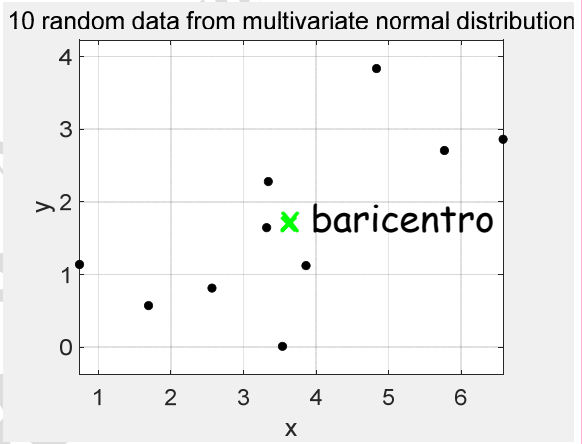
la stessa retta



Rette di regressione in $\mathbb{R}^2$ con MATLAB

```
%% P: 2D punti campione da N(μ,Σ)
N=10; MU=[3 1]; SIGMA=[1 .2; .2 .7];
P=mvnrnd(MU,SIGMA,N);
B0=mean(P); % baricentro (media campionaria)
Xi=P(:,1); Yi=P(:,2);
plot(Xi,Yi,'ok','MarkerFaceColor','k')
axis equal; grid on
```

$$\Sigma = \begin{pmatrix} 1 & 0.2 \\ 0.2 & 0.7 \end{pmatrix}$$



regressione di y su x

$$r : y = ax + b$$

$$y_i = ax_i + b, \quad i = 1,\ldots,n$$

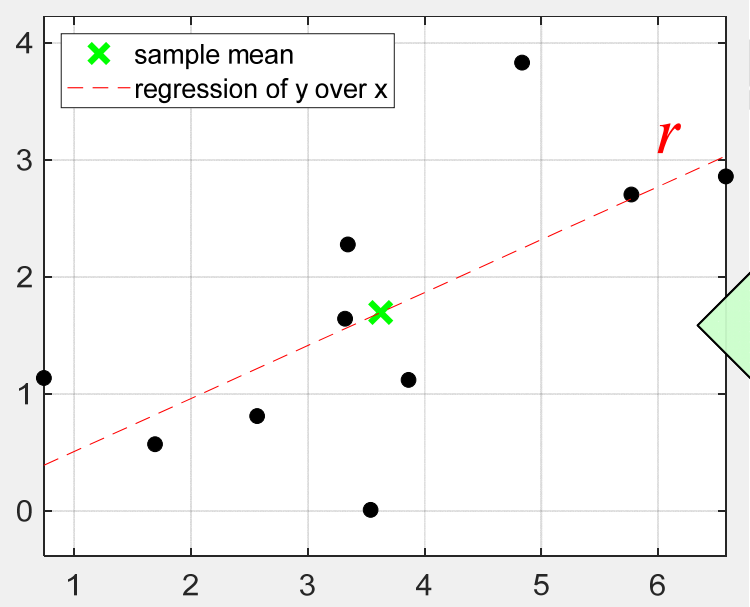
$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

regressione di x su y

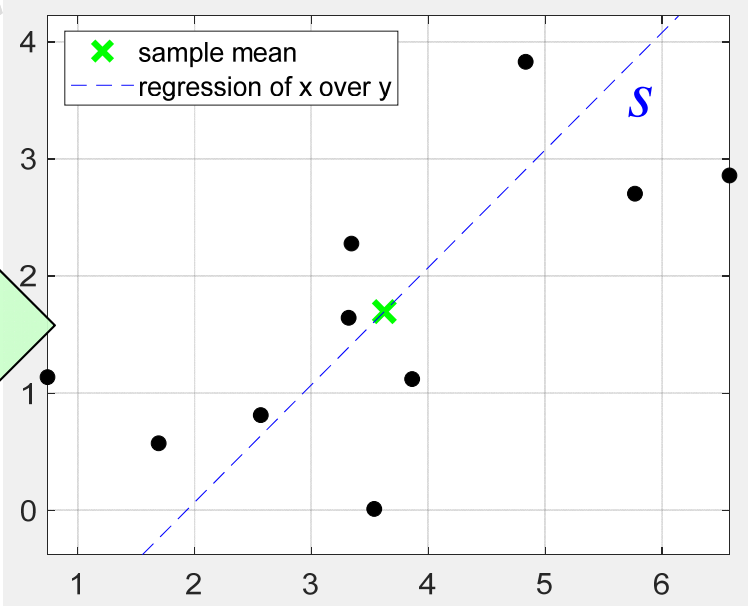
$$s : x = cy + d$$

$$x_i = cy_i + d, \quad i = 1,\ldots,n$$

$$\begin{pmatrix} y_1 & 1 \\ y_2 & 1 \\ \vdots & \vdots \\ y_n & 1 \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$



rette diverse



Rette di regressione in $\mathbb{R}^2$ con MATLAB

regressione di y su x

$$r: y = ax + b$$

$$y_i = ax_i + b, \quad i = 1,\ldots,n$$

$$J_{LS}^{(1)}(a,b) = \sum_{i=1}^{n}\left[y_i - (ax_i + b)\right]^2$$

regressione di x su y

$$s: x = cy + d$$

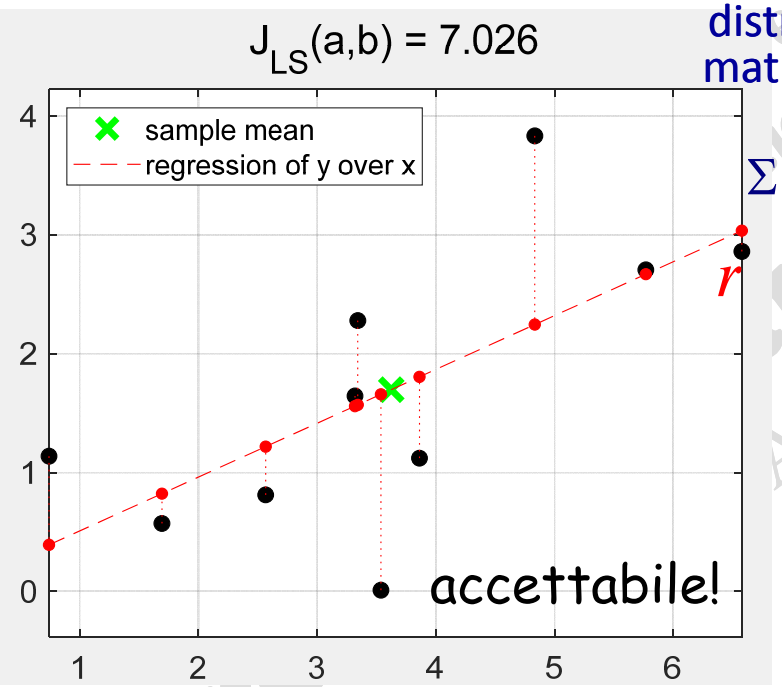
$$x_i = cy_i + d, \quad i = 1,\ldots,n$$

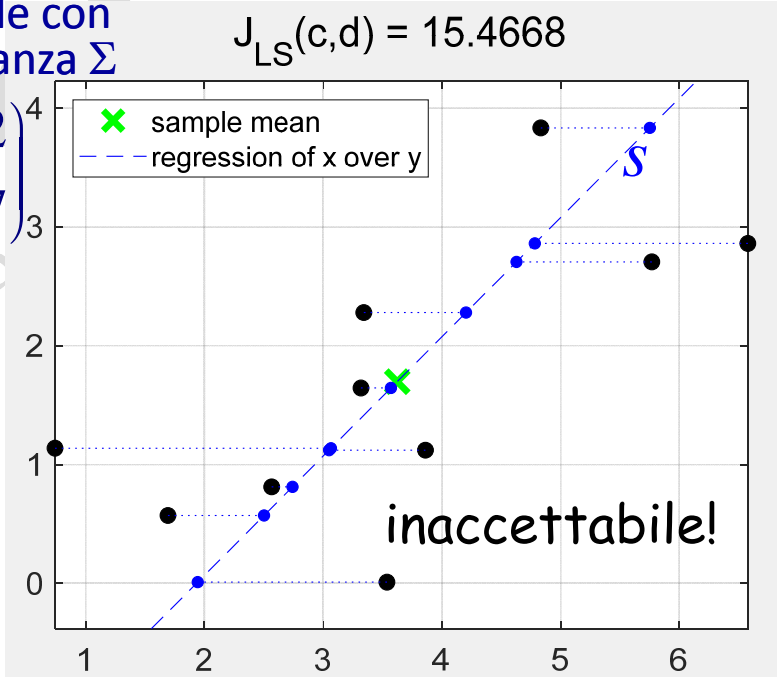
$$J_{LS}^{(2)}(c,d) = \sum_{i=1}^{n}\left[x_i - (cy_i + d)\right]^2$$

Stesso campione di dati da distribuz. Normale con matrice di covarianza Σ

$$\Sigma = \begin{pmatrix} 1 & 0.2 \\ 0.2 & 0.7 \end{pmatrix}$$

N=10





Esercizio: produrre grafici simili per N=20 con i valori dei J_{LS}

Piano Least Squares in $\mathbb{R}^3$

Per definizione il piano "Least Squares" $\pi : z=ax+by+c$, relativo ai dati $(x_i,y_i,z_i)_{i=1,\dots,n}$, minimizza la somma dei quadrati dei residui: min $J_{LS}(a,b,c)$

$$J_{LS}(a,b,c)=\sum_{i=1}^{n}\underbrace{\left[z_i-(ax_i+by_i+c)\right]}_{\text{residui}}^2$$

$$J_{LS}(a,b,c)=\left\|\begin{pmatrix}x_1 & y_1 & 1\\ x_2 & y_2 & 1\\ \vdots & \vdots & \vdots\\ x_n & y_n & 1\end{pmatrix}\begin{pmatrix}a\\ b\\ c\end{pmatrix}-\begin{pmatrix}z_1\\ z_2\\ \vdots\\ z_n\end{pmatrix}\right\|_2^2=\left\|A\begin{pmatrix}a\\ b\\ c\end{pmatrix}-\mathbf{z}\right\|_2^2$$

Come già visto per la retta Least Squares, anche il **piano Least Squares**, per i dati $(x_i,y_i,z_i)_{i=1,\dots,n}$, passa per il punto media campionaria

$$P_\mu\equiv(\bar{x},\bar{y},\bar{z})=\left(\frac{1}{n}\sum_{i=1}^{n}x_i,\frac{1}{n}\sum_{i=1}^{n}y_i,\frac{1}{n}\sum_{i=1}^{n}z_i\right)$$

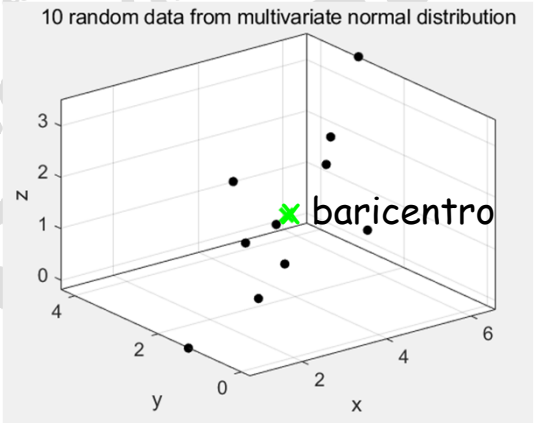
In generale l'iperpiano Least Squares in $\mathbb{R}^d$, di dimensione d–1 e relativo agli n dati ($n>d$) $P_i\left(x_1^{(i)},x_2^{(i)},\dots,x_d^{(i)}\right)_{i=1,\dots,n}$ minimizza il funzionale J_{LS}

Piano di regressione di z su x,y in $\mathbb{R}^3$ con MATLAB

(piano Least Squares) $\pi_z : z = ax + by + c$

Campione di dati da distrib. Normale con matrice di covarianza Σ

$$\Sigma = \begin{pmatrix} 1 & 0.2 & 0.7 \\ 0.2 & 1 & 0 \\ 0.7 & 0 & 1 \end{pmatrix}$$



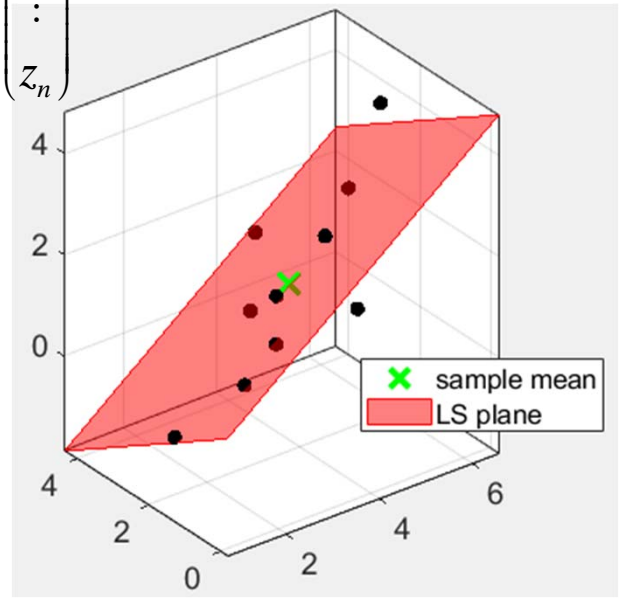

```
%% P: punti campione 3D da N(μ,Σ)
N=10; P=mvnrnd([3 1 1],[1 .2 .7; .2 1 0; .7 0 1],N);
B0=mean(P); % media campionaria (baricentro)
Xi=P(:,1);  Yi=P(:,2);  Zi=P(:,3);
```

2° metodo (più generale)

$$z_i = ax_i + by_i + c, \quad i = 1,\ldots,n$$

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}$$

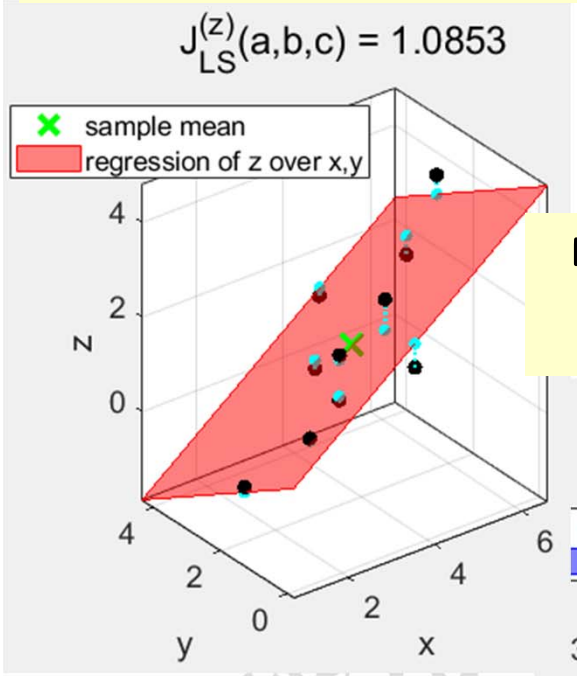
```
%% piano Least Squares
A=[Xi Yi ones(N,1)];
coefZ = A\Zi;
```



Piani di regressione in $\mathbb{R}^3$ con MATLAB

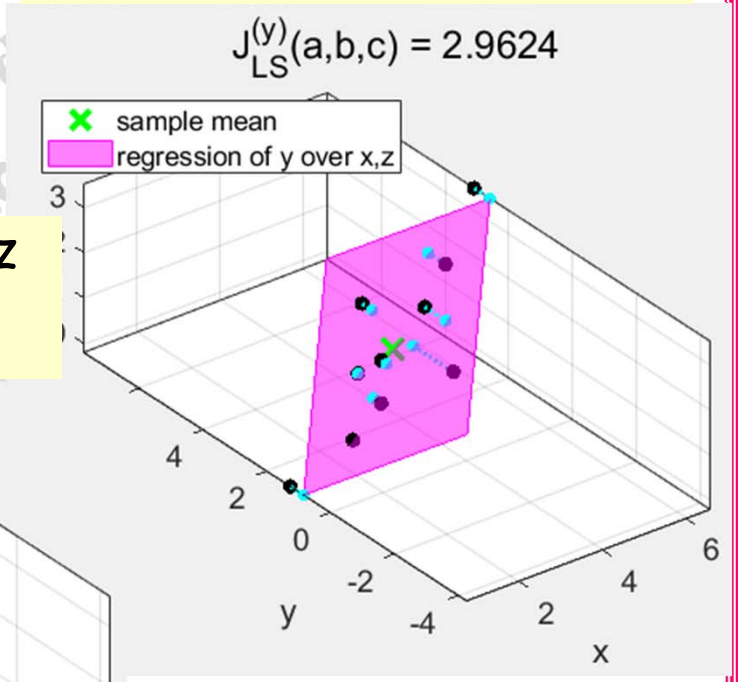
regressione di z su x,y

$\pi_z : z = ax + by + c$



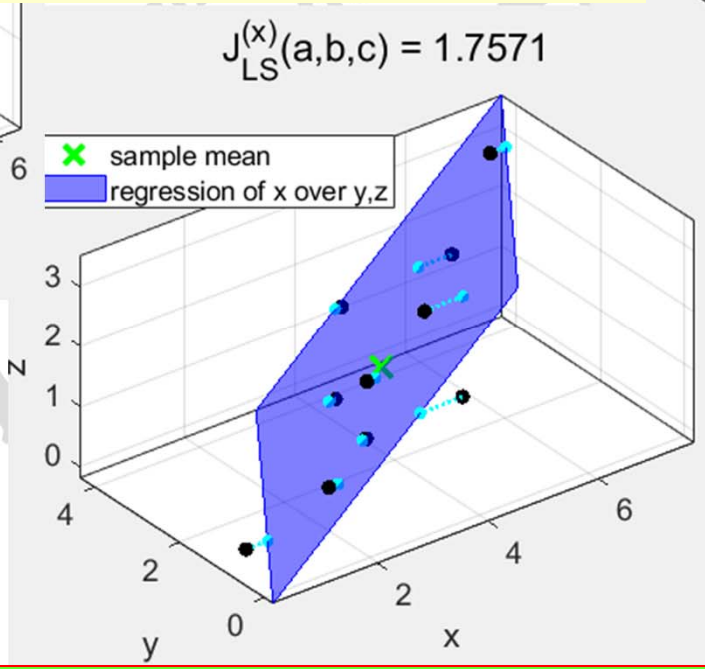
regressione di y su x,z

$\pi_y : y = ax + bz + c$



regressione di x su y,z

$\pi_x : x = ay + bz + c$



Esercizio: produrre grafici simili per N=20 con il valore di J_{LS}

Retta Least Squares VERSUS retta PCA

dati di input: $P_i\left(x_i, y_i\right)_{i=1,\ldots,n}$

media campionaria $P_\mu(\bar{x},\bar{y})$:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \text{mean}(\vec{x})$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i = \text{mean}(\vec{y})$$

- La retta Least Squares $y=ax+b$ minimizza il funzionale J_{LS}:

$$J_{LS} = \left\| A\begin{pmatrix} a \\ b \end{pmatrix} - \mathbf{y} \right\|_2^2$$

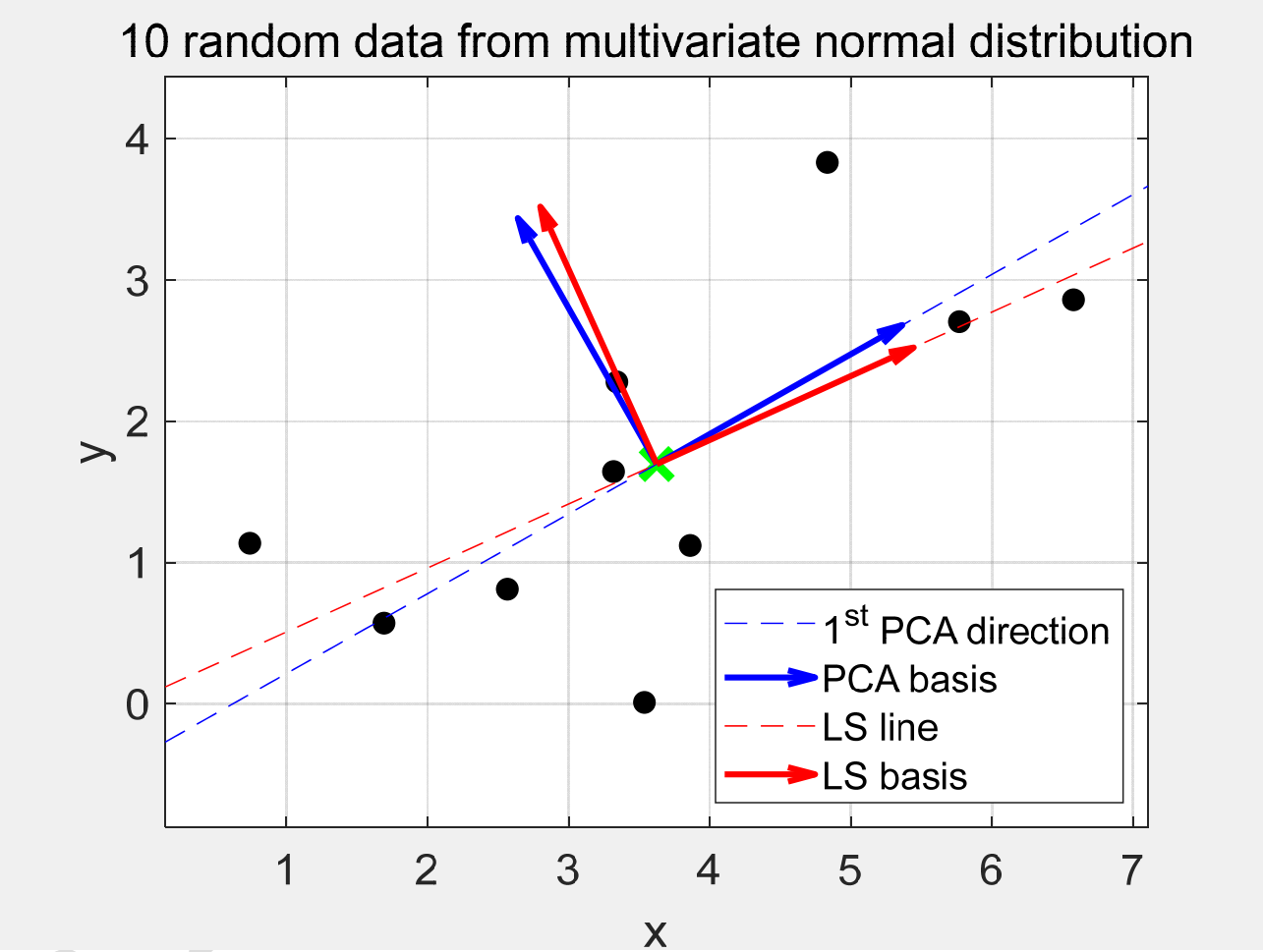
dove $A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$

- La retta 1ª direzione principale PCA minimizza il funzionale J_{PCA}:

$$J_{PCA} = \sum_{i=1}^{n} \left\| P_i - P_i^{PCA} \right\|_2^2$$

dove P_i^{PCA} è la proiezione ortogonale di P_i sulla retta passante per P_μ e parallela a **e**, che è l'autovettore relativo al massimo autovalore della matrice di covarianza campionaria.

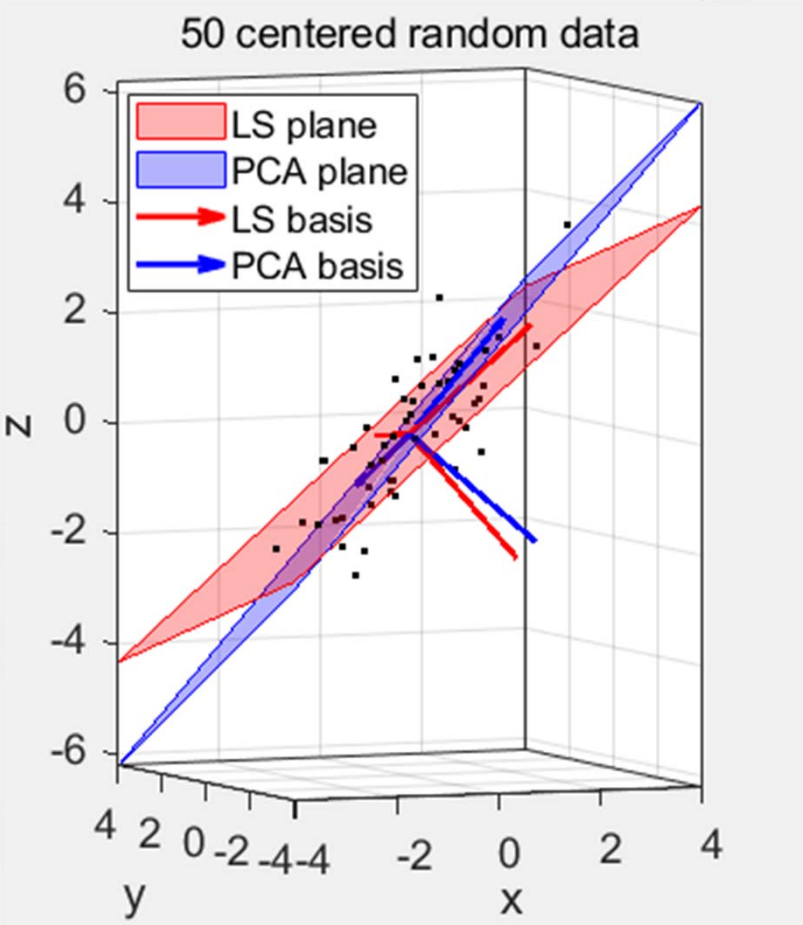
Retta Least Squares VS retta PCA



Perché le rette Least Squares e PCA sono diverse?
Quale condizione ottimale soddisfa ciascuna retta rispetto ai campioni? Come verificarlo?

Analogamente:

Piano Least Squares VS piano PCA



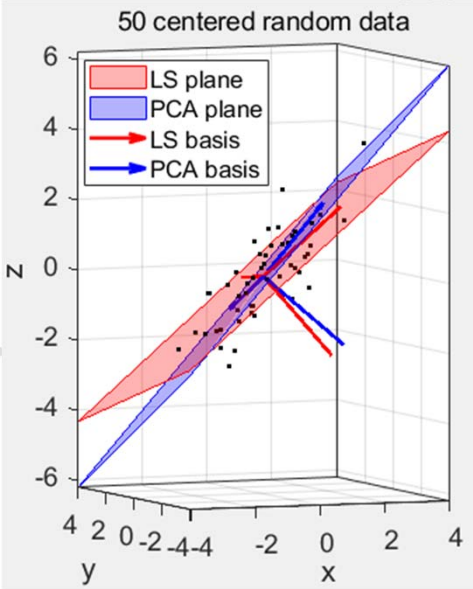
Perché I piani Least Squares e PCA sono diversi?
Quale condizione ottimale soddisfa ciascun piano rispetto ai campioni? Come verificarlo?

Piano Least Squares VS piano PCA

```
rng('default'); N=50; P=mvnrnd([0 0 0],[1 .2 .7; .2 1 0; .7 0 1],N); % N 3D points
B0=mean(P); Xc=P-repmat(B0,size(P,1),1); % centered data
[base, comp, lambda] = pca(P);
plot3(P(:,1),P(:,2),P(:,3),'k.'), axis equal; hold on
% Piano PCA: piano passante per B0 e generato da base(:,1:2)
%                    i coefficienti del piano sono dati da base(:,3)
AX=axis; Px=[AX(1:2);AX(1:2)]; Py=[AX(3:4);AX(3:4)]';
Pz=B0(3)-(base(1,3)*(Px-B0(1))+base(2,3)*(Py-B0(2)))/base(3,3); h=surf(Px,Py,Pz); set(h, …
quiver3(B0(1)*ones(1,3),B0(2)*ones(1,3),B0(3)*ones(1,3),base(1,:),base(2,:),base(3,:),0)
```

o equivalentemente

```
% Piano PCA: piano passante per B0 e generato da base(:,1:2)
syms a b real;  plane=bas(:,1:2)*[a;b];
hPCA=ezsurf(B0(1)+plane(1),B0(2)+plane(2),B0(3)+plane(3),[-3 3]);
set(hPCA,'EdgeColor','none','FaceColor','b','FaceAlpha',0.5)
```



Esercizio

Oltre al piano PCA (le prime due direzioni principali), tracciare il (piano LS di z risp. A x,y) per lo stesso campione random.

Quale condizione ottimale soddisfa ciascun piano rispetto ai campioni? Come verificarlo?