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

DiST Dipartimento di Scienze e Tecnologie

L. Magistrale in IA (ML&BD)

Scientific Computing (part 2 – 6 credits)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Bldg. C4
room: n. 423 – North Side, 4th floor
phone: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

Contents

- **Outlines of inferential statistics:**
  - **Data matrix.**
  - **Sample mean.**
  - **Centered data matrix and standardized matrix.**
  - **Scatter matrix.**
  - **Covariance matrix.**
  - **Correlation matrix.**

… from statistical point of view

and … from Linear Space point of view

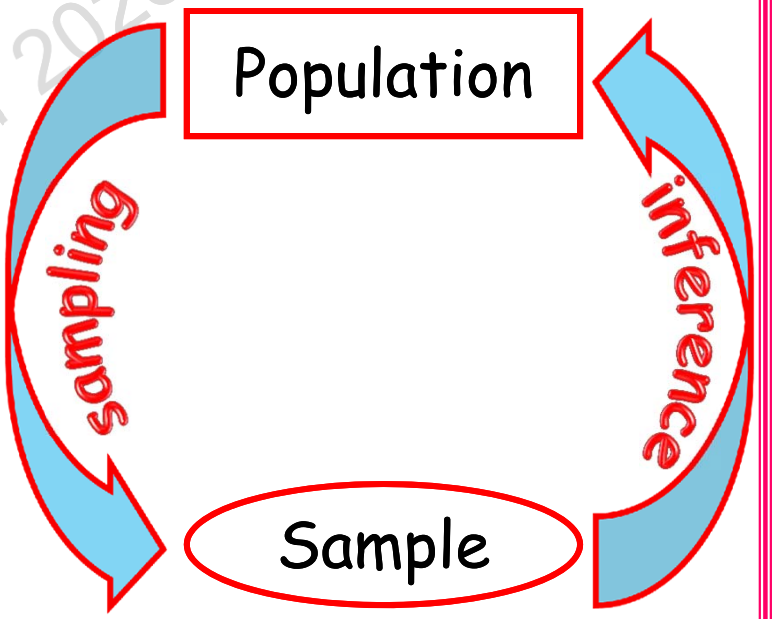
Outline of Inferential Statistics

Statistics is a quantitative science, and it is the right tool for making decisions in situations of uncertainty.

Inferential Statistics is a field of statistics that uses analytical tools for drawing conclusion about a population by examining random samples. The goal of Inferential Statistics is to make generalizations about a population.

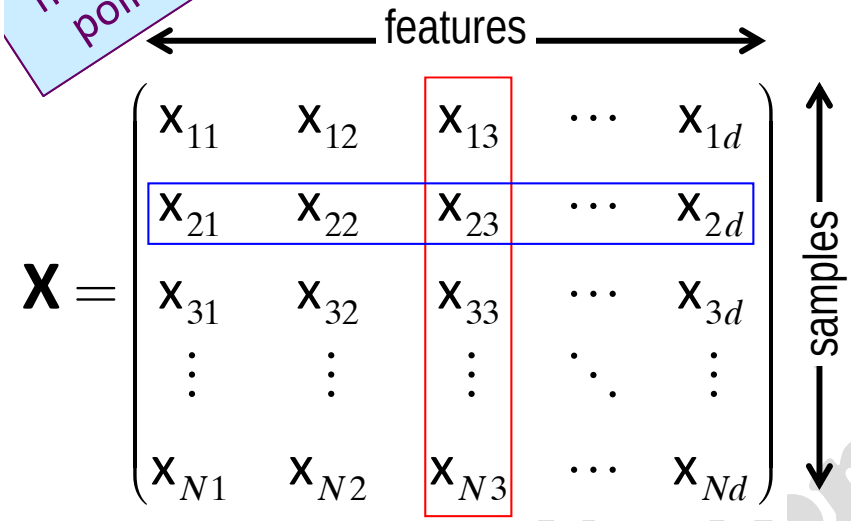
Statistical Inference is carried out according to the following steps:

- Extract a part of the population (sampling).
- Compute some quantities (estimates) from sample data such as, for example, mean, variance or others.
- Extend the results provided by the sample to the population (inference).



Data matrix X

from **statistical** point of view



X is a matrix of size $N \times d$, where:

- N is the number of *samples* (*data*).
- d is the number of *random variables* (*r.v.*), or *features*, and also the dimension of the data Space.

Every row in **X** contains a *sample* (*datum*).

Every column in **X** contains a *random variable* (*feature*).

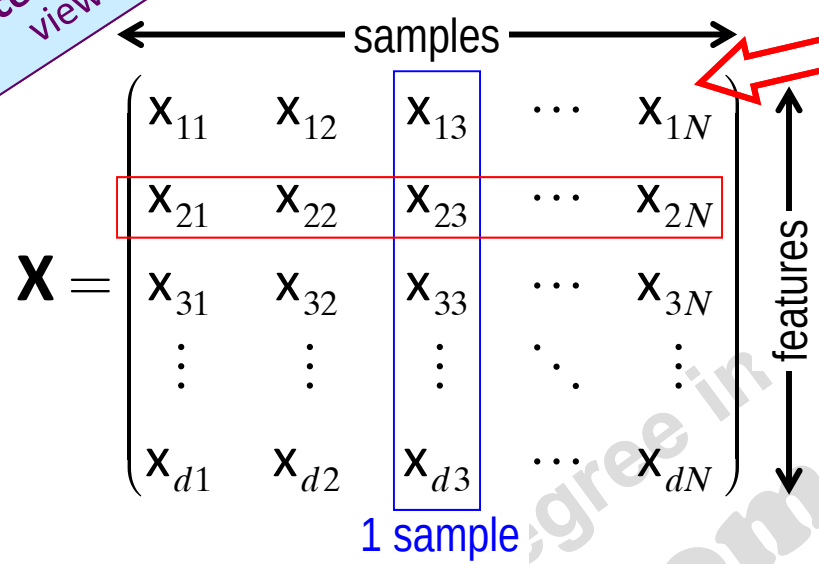
x_{ij} is the *variable* (*feature*) #j picked from the *sample* #i.

```
X=[ -1     1     2     2
    -2     3     1     0
     4     0     3    -1 ];
N=size(X,1); % number of samples
d=size(X,2); % number of random variables
```

Statistical softwares expect the data matrix in this format.

Data matrix X

from **Linear Space** point of view



transpose of the previous matrix

X is a matrix of size $d \times N$, where:

- N is the number of *samples* (*data*).
- d is the number of *random variables* (*features*).

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

Every row in **X** contains a *random variable* (*feature*).

Every column in **X** contains a *sample* (*datum*).

x_{ij} is the *variable* (*feature*) #i picked from the *sample* #j.

This notation is closer to Matrix Algebra, because *samples* are considered as column vectors, and their components are the *r.v.s*. This means that the *N* samples belong to a *d*-dimensional space (*feature space*).

Numerical softwares expect the data matrix in this format.

from **Linear Space** point of view

columns: samples

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

rows: features

The *sample mean* vector $\boldsymbol{\mu}$, has size ($d \times 1$), and it is the column vector whose components are computed as a mean of values of each row (i.e. the mean value of each *feature* over all the samples):

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

The *centered data matrix* $\mathbf{X}_C$ is obtained by subtracting the *sample mean vector* $\boldsymbol{\mu}$ from each sample, i.e. from each column of $\mathbf{X}$:

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

where $\mathbf{X}_C = \mathbf{X} - \boldsymbol{\mu}\, \mathbf{1}_N^T$ (outer product)

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

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

from **Linear Space** point of view

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

columns: samples

rows: features

By means of the centered data matrix $\mathbf{X}_C$ we can compute:

The **Scatter matrix S** ($d \times d$): the sum of the outer products of centered samples $(\mathbf{x}^{(k)} - \boldsymbol{\mu})$:

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

The **Covariance matrix C** ($d \times d$):

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

The **Correlation matrix R** ($d \times d$):

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

where $\mathbf{X}_S = \mathbf{D}^{-1}\mathbf{X}_C$ is the *standardized matrix*

and $\mathbf{D} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d)$ σ_j : std. dev. of jth *r.v.*.

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

from **Linear Space** point of view

MATLAB example

$\mathbf{X}$

variables	samples X_1	X_2	X_3
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);      % number of samples
mu  = mean(X,2);    % column vector of means
sig = std(X,0,2);   % vector of std. deviations.
```

```
std(X,0)  standard deviation normalized by 1/(N-1)   default
std(X,1)  standard deviation normalized by 1/N
```

centered matrix $\mathbf{X}_C$

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

zero mean

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

standardized matrix $\mathbf{X}_S$

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

zero mean

```
std(Xs,0,2)
ans =
     1
     1
     1
     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  % without inverse
ans =
   -0.41478   -0.72587    1.1406
   -0.21822    1.0911    -0.87287
          0        -1          1
     1.0911   -0.21822   -0.87287
```

from Linear Space point of view

MATLAB example

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % number of samples
mu = mean(X,2); % column vector of means
```

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

MATLAB functions **cov()**, **corrcoef()**, **pca()**, … want, as a parameter, the data matrix **X** as from a statistical point of view, i.e. of size: $N \times d$ (samples × features)
and **not** $d \times N$ (features × samples)

We need to pass the transpose of **X**.

```
C=cov(X')   % covariance matrix
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
```

centered matrix $\mathbf{X}_C$

```
Xc = X - repmat(mu,1,N);
Xc*Xc'/(N-1) % scaled scatter matrix
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') % correlation matrix
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
```

standardized matrix $\mathbf{X}_S$

```
Xs = normalize(X,2); % standardiz. matrix
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
```

from Linear Space point of view

MATLAB example: PCA*

* in MATLAB Statistics and Machine Learning Toolbox

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

variables

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % number of samples
mu = mean(X,2); % column vector of means
```

← All the output parameters →

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

scores (λ)

Hotelling's T-squared statistic

percentage of total variance

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

percentages

data reconstruction

```
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
```

```
basis*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
```

data projection

```
basis'*Xc % =basis\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
```

from Linear Space point of view

MATLAB example: PCA and covariance matrix

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

```
X = [-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % number of samples
mu = mean(X,2); % column vector of means
```

```
[basis,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)   scaled eigenvalues
ans = logical   1
all(all(Vscat == V))    the same eigenvectors
ans = logical   1
```

```
V
```

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

```
basis
```

```
basis =
      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) basis(:,3:4)])
ans =
     2
```

linearly dependent

from Linear Space point of view

MATLAB example: PCA and SVD (singular value decomposition)

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

variables

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

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % number of samples
mu = mean(X,2); % column vector of means
```

```
[basis,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
```

```
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
```

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

`svd(A,'econ')` A(m×n)

- m > n : only the first n columns of U are computed, and S is n-by-n.
- m = n : svd(A,"econ") is equivalent to svd(A).
- m < n : o nly the first m columns of V are computed, and S is m-by-m.

```
basis
```

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

dimensionality reduction

among the 4 dimensions of the random variables (features), just two are sufficient to describe all the data.

from Linear Space point of view

MATLAB example: PCA and SVD

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

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % number of samples
mu = mean(X,2); % column vector of means
```

```
[basis,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) is equivalent to svd(A,"econ").
m ≤ n : svd(A,0) is equivalent to svd(A,"econ").

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
```

basis

```
basis =
      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
```

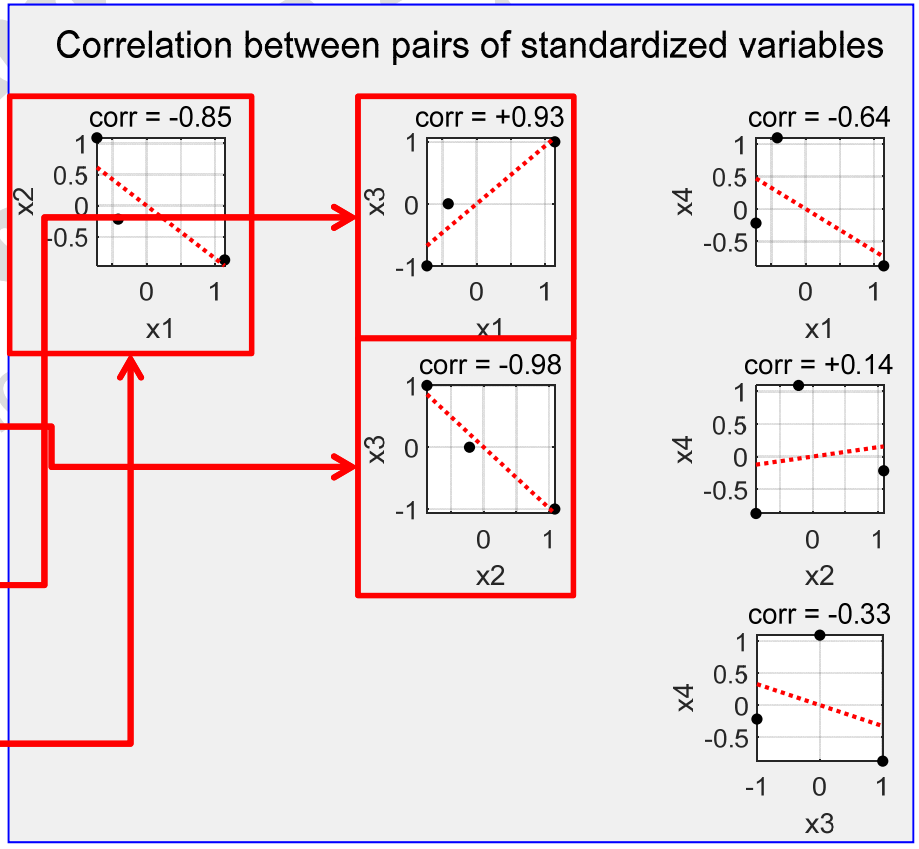
dimensionality reduction

among the 4 dimensions of the random variables (features), just two are sufficient to describe all the data.

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N = size(X,2);  % number of samples
mu = mean(X,2); % column vector of means
```

```
R=corrcoef(X')  % correlation matrix
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');  % standardiz. matrix
```



High correlation between:

- 2nd and 3rd variables — negative correlation
- 1st and 3rd variables — positive correlation
- 1st and 2nd variables — negative correlation

Recap

from **statistical** point of view

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1];
N=size(X,1); % number of samples
mu=mean(X);  % row vector of means
Xs=normalize(X);        % standardized matrix
Xc=X-repmat(mu,N,1); % centered matrix
[Bx,Cx,Lx]=pca(X,'Economy',false);
[Bc,Cc,Lc]=pca(Xc,'Economy',false);
[Bs,Cs,Ls]=pca(Xs,'Economy',false);
```

from **Linear Space** point of view

```
X=[-1 1 2 2; -2 3 1 0; 4 0 3 -1]';
N=size(X,2);  % number of samples
mu=mean(X,2); % col vector of means
Xs=normalize(X');      % standardized matrix
Xc=X-repmat(mu,1,N); % centered matrix
[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
```

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

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
```

Contents

- What is PCA?
- PCA Algorithm derivation.
- PCA Algoritms.
- Geometrical interpretation of PCA.

Principal Component Analysis (PCA) appears in many fields of applied mathematics under different names (Karhunen-Loeve Transform, Hotelling Transform, …).

PCA was created in 1901 by Karl Pearson, but Harold Hotelling indipendently developed and named the Principal Component Analysis method in 1933.

What is PCA?

PCA is a procedure that transforms (possibly) correlated variables into uncorrelated variables called principal components.

Why is PCA used?

For data "dimensionality reduction", i.e. to compress data size with (almost) no loss of information.

How does PCA work?

PCA looks for directions with maximal data variation.

The 1st principal direction is located by an eigenvector related to the maximum eigenvalue of the data Covariance Matrix. Then similarly for the 2nd ...

Moreover, PCA projects data on these directions, obtaining their principal components.

For PCA derivation

Recall:

Gradient (∇) properties of the standard scalar product

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

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

$\boldsymbol{v}$ does not depend on $\mathbf{x}$

similar to: $f(x)=\alpha x \Rightarrow f'(x)=\alpha$

PROOF:

j^{th} component of the gradient $\frac{\partial}{\partial x_j} F = v_j$

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

similar to: $f(x)=x^2 \Rightarrow f'(x)=2x$

PROOF:

j^{th} component of the gradient $\frac{\partial}{\partial x_j} F = 2x_j$

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

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

$\mathbf{A}$ is symmetric and does not depend on $\mathbf{x}$

For PCA derivation

Recall:

Gradient (∇) properties of the standard scalar product

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

$\mathbf{A}$ is symmetric and does not depend on $\mathbf{x}$

PROOF:

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

$(\nabla_{\mathbf{x}} F)_j$: j[th] component of the gradient of F:

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

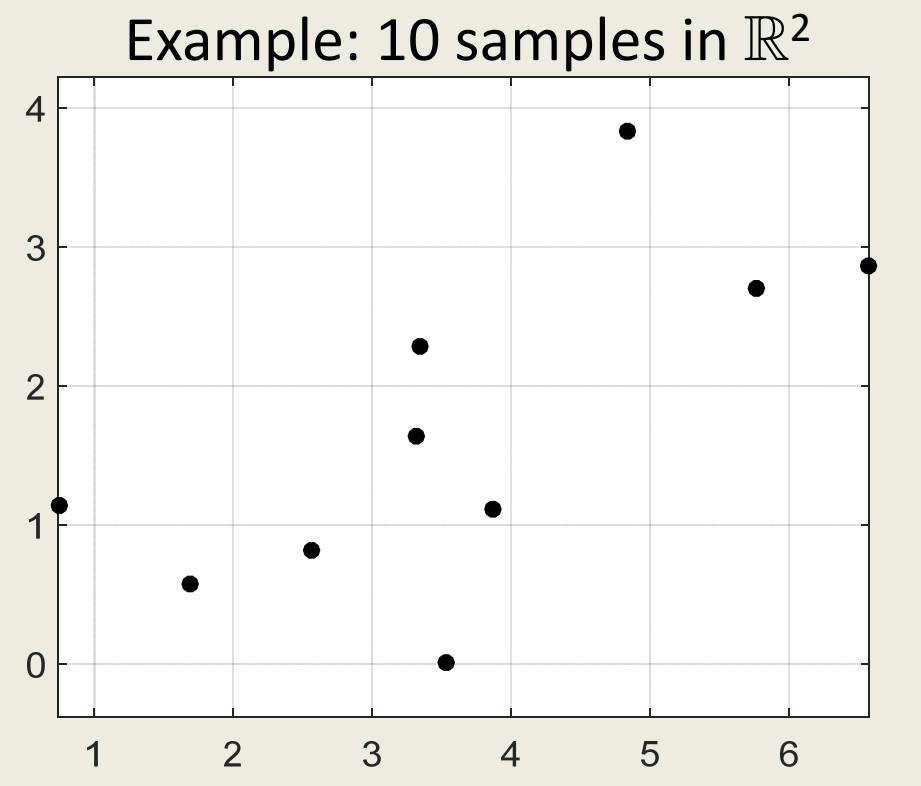
$$= y_j(\mathbf{x}) + \sum_{i=1}^{d} x_i \frac{\partial}{\partial x_j} \sum_{k=1}^{d} A_{ik} x_k = y_j(\mathbf{x}) + \sum_{i=1}^{d} x_i \sum_{k=1}^{d} \frac{\partial}{\partial x_j} \{A_{ik} x_k\} =$$

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

since $\mathbf{A}$ is symmetric

PCA derivation

Let us consider N samples $\mathbf{x}^{[k]} \in \mathbb{R}^d$.



We look for a subspace of the *Affine Space* $\mathbb{R}^d$, of dimension $K << d$, that meaningfully describes all the data. Its name is "reduced" PCA subspace.

PCA derivation (cont.)

PCA subspace of dimension 0 (a single point): we want to describe data by means of a point **c**, such that "it minimizes the sum of its euclidean distances from all the samples", i.e.

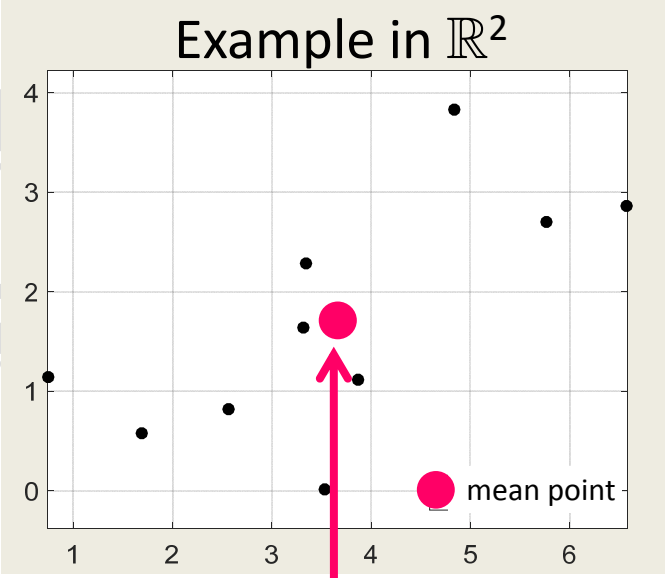
Find **c** that minimizes J_0 : $J_0 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2$

sum of squared euclidean norms

⇩

convex function*

* any norm is always a convex function; the same holds for the sum of convex functions



c must be the sample mean **m**.

PCA derivation (cont.)

The point c that **minimizes** $J_0 = \sum_{k=1}^{N} \left\| \mathbf{x}^{[k]} - \mathbf{c} \right\|_2^2$ is the sample mean m: Why?

PROOF:

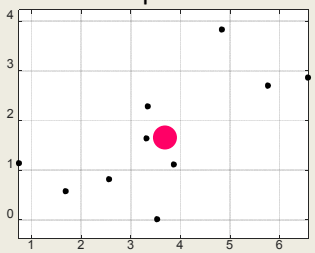
By properties of gradients of the std. scalar products and its induced norms:

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

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

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

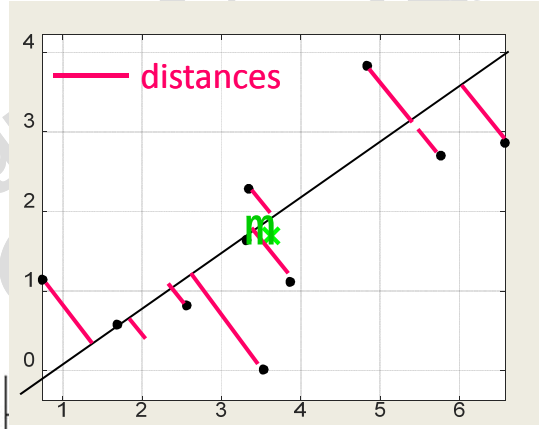
Example in $\mathbb{R}^2$

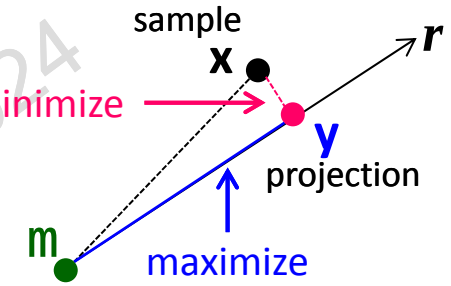


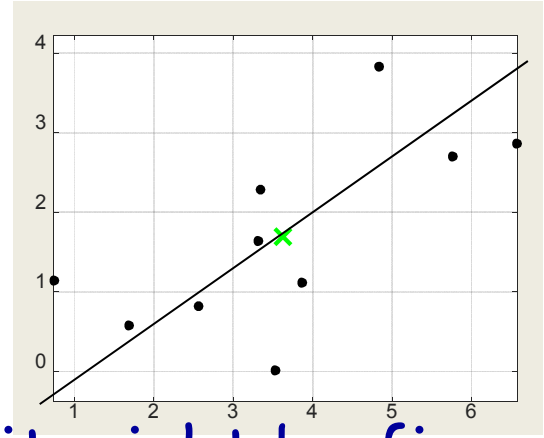
Nevertheless, the sample mean m alone is not able to describe sufficiently all the data.

Incremental PCA derivation (cont.)

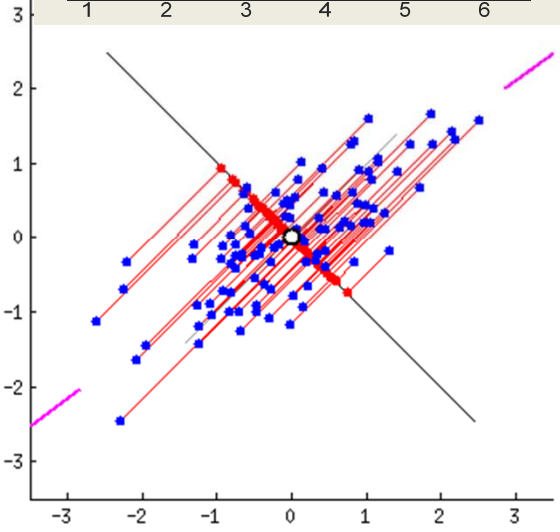
Then, we look for a line (PCA subspace of dimension 1) such that "it minimizes the sum of its euclidean distances from all the samples", i.e. minimizing the sum of projection residuals

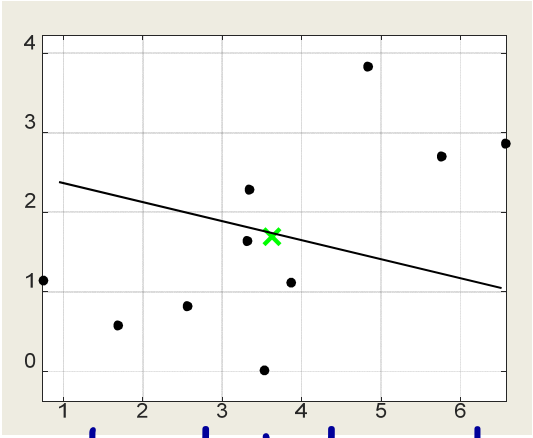






it might be fine ...





absolutely no!

Incremental PCA derivation (cont.)

In order to find the 1st *principal direction*, we look for a line *r*, in the affine space $\mathbb{R}^d$, passing through a point **c** and parallel to a normalized vector **e** ($\|\mathbf{e}\|_2=1$):

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

such that "*r* minimizes the sum of euclidean distances between all the samples $\mathbf{x}^{[k]}$ and their orthogonal projections $\mathbf{y}^{[k]}$ on the line"

projection matrix on the direction space $\mathbf{P}: \mathbf{v}^{[k]} = \mathbf{x}^{[k]} - \mathbf{c} \longrightarrow \mathbf{w}^{[k]} = \mathbf{P}\,\mathbf{v}^{[k]} = \mathbf{y}^{[k]} - \mathbf{c} = \alpha^{[k]}\mathbf{e}$

P acts on the Linear Space of vectors starting from **c** (as Origin)

Find **c**, **e** and $\alpha^{(k)}$ such that

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

Reconstruction squared error (SSE: sum of squared errors)

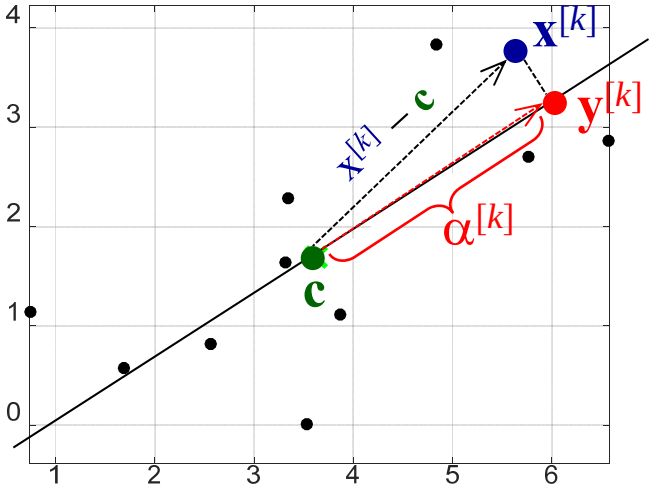
Incremental PCA derivation (cont.)

$(\mathbf{y}^{[k]}-\mathbf{c})$ is the orthogonal projection of $(\mathbf{x}^{[k]}-\mathbf{c})$ on the line $r \parallel \mathbf{e}$

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

To get these vectors, all the data points were centered on c

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



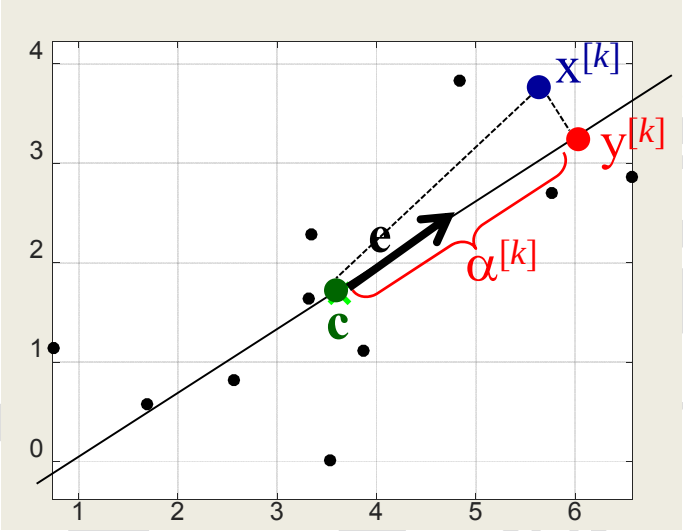
To minimize J_1, the scalar $\alpha^{[k]}$ must be the component of the vector $(\mathbf{x}^{[k]} - \mathbf{c})$ along r, i.e.:

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

where $\langle , \rangle$ is the standard scalar product inducing the euclidean norm.

Incremental PCA derivation (cont.)

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



By hypothesis: $\|\mathbf{e}\|_2=1$

Applying some properties of the standard scalar product and its induced norm, we get:

expand it as the square of a binomial $(A-B)^2 = A^2 + B^2 - 2AB$

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

$\alpha^{(k)}$

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

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

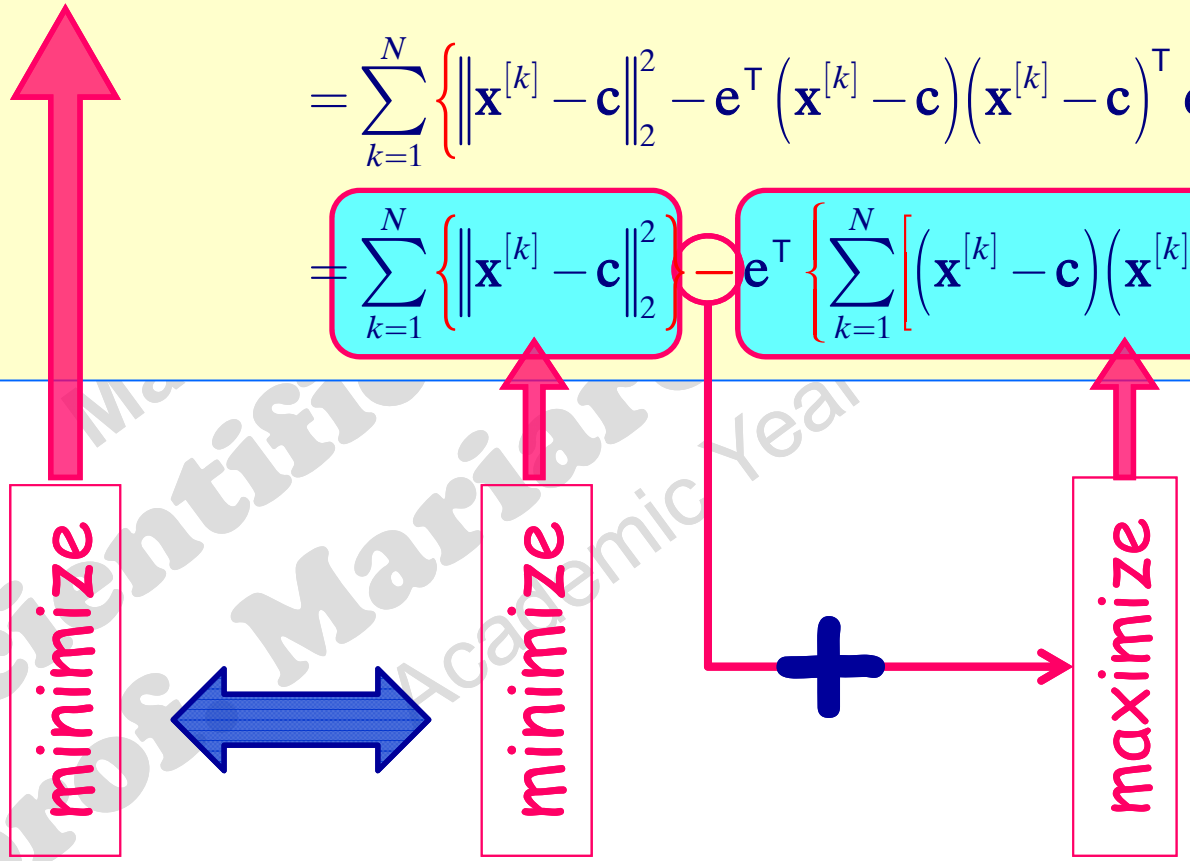
Incremental PCA derivation (cont.)

Putting the result in J_1, we have:

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

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

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



Incremental PCA derivation (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 $\Rightarrow$ this is the **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 is proportional to the covariance matrix of data $\mathbf{x}^{[k]}$.

minimize

minimize

maximize

c must be the *sample mean* **m** of the data $\mathbf{x}^{[k]}$

e is the direction that maximizes data variance

PCA derivation (cont.)

Constrained Optimization Problem

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

subject to the equality constraint: $\mathbf{e}^T\mathbf{e} = 1$.

A constrained optimization problem with only equality constraints can be solved by the method of Lagrange multipliers, by which we seek the stationary points of the Lagrangian function $\mathbb{L}$:

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

They are the zeros of the gradient ∇ of $\mathbb{L}$().

Incremental PCA derivation (cont.)

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

$\frac{\partial}{\partial\lambda}\mathbb{L}(\mathbf{c},\mathbf{e},\lambda)=0$ $\Longleftrightarrow$ $\mathbf{e}^{\mathsf{T}}\mathbf{e}=1$ (true by hypothesis)

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

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

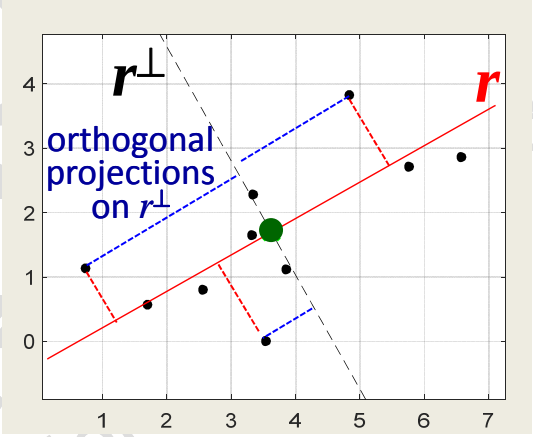
$\Longleftrightarrow$ $\boxed{S\mathbf{e}=\lambda\mathbf{e}}$ (typical eq. for eigenvalues/eigenvectors) i.e.:

- **e** is an eigenvector related to the eigenvalue λ of the *Scatter matrix* S (**e** is also eigenvector of the *Covariance matrix*);
- λ is the maximum eigenvalue, since min J_1 is equivalent to:

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

Incremental PCA algorithm

In the linear space (centered at **m**), once the first principal direction has been found (r : r=span{**e**}), in order to seek the second principal direction, we consider $r^{\perp}$ (the orthogonal complement of r) and repeat the same previous procedure on orthogonal projections of data $\mathbf{x}^{[k]}$ on $r^{\perp}$.

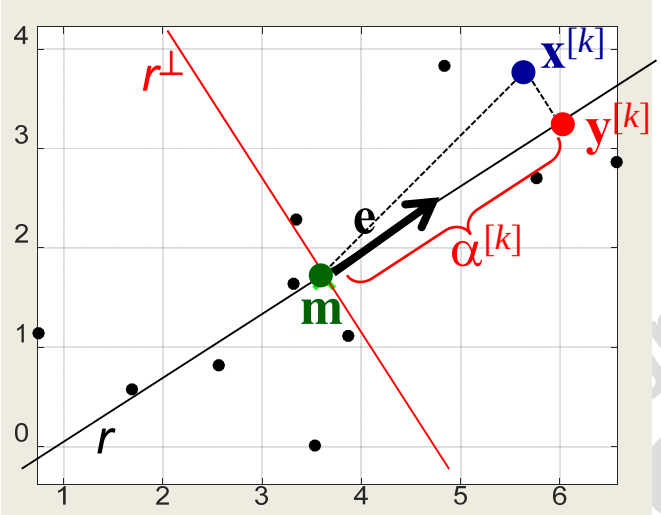


Remember that $\mathbb{R}^d = r \oplus r^{\perp}$ and $r \perp r^{\perp}$: then $r^{\perp}$ is a subspace of dimension d−1. The procedure can be repeated up to d=1.

How to project orthogonally onto $r^{\perp}$?

How to project orthogonally onto $r^{\perp}$?

$(\mathbf{y}^{[k]}-\mathbf{m})$ is the orthogonal projection of $(\mathbf{x}^{[k]}-\mathbf{m})$ onto r



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

$\mathbf{P}$ is the *orthogonal projection matrix* onto r=span{$\mathbf{e}$} ($\mathbf{e}$: orthonormal basis):

simplified formula $\Rightarrow$ $\mathbf{P} = \mathbf{e}\,\mathbf{e}^{\top}$

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

$\Rightarrow$ $\mathrm{I} - \mathrm{P}$ is the orthogonal projection matrix on $r^{\perp}$

$\mathrm{I} - \mathrm{P}$ is said the "complementary projection matrix"

Proof: In facts, the vector $\mathbf{x}^{[k]} - \mathbf{y}^{[k]}$, orthogonal to r, can be written as:

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

Example: PCA in MATLAB

Statistics and Machine Learning Toolbox

Random sample from the Multivariate Normal Distribution

```
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);
```

basis of $\mathbb{R}^d$

principal components

eigenvalues of cov(X)

features

samples

data matrix **X**

from **statistical** point of view

X of size (N×d) where:

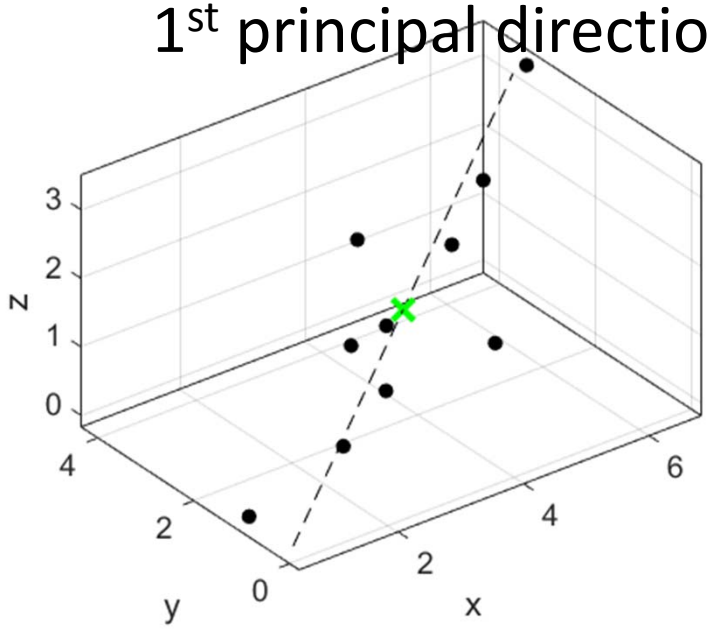
N is the number of samples,

d is the dimension of data space

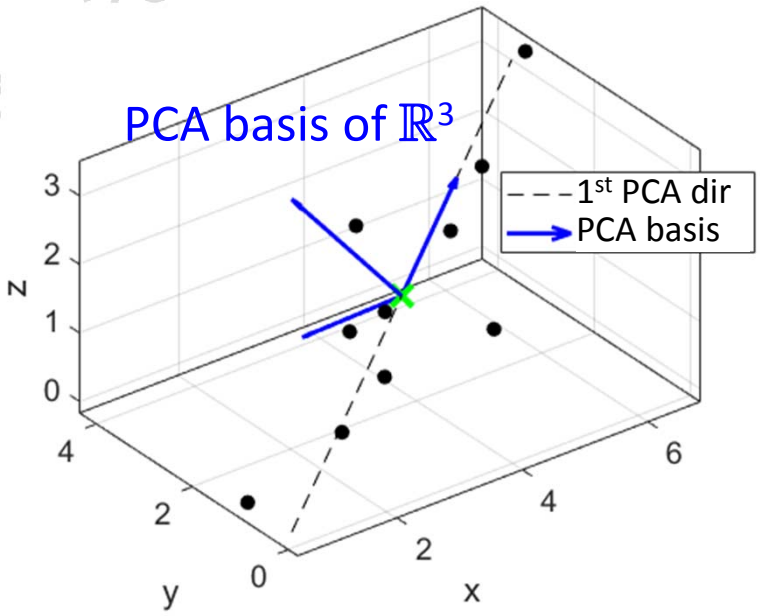
`'Economy',false`:

to get a full basis of $\mathbb{R}^d$

1st principal direction



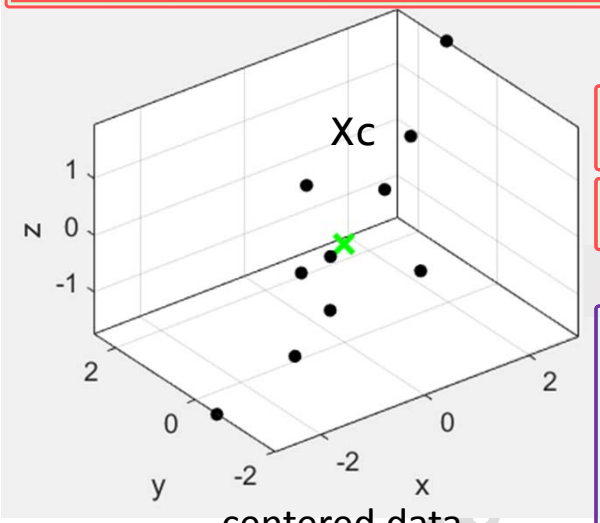
PCA basis of $\mathbb{R}^3$



Incremental PCA algorithm illustration (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);
```

from **statistical** point of view



centered data

basis of *r* (**e**)

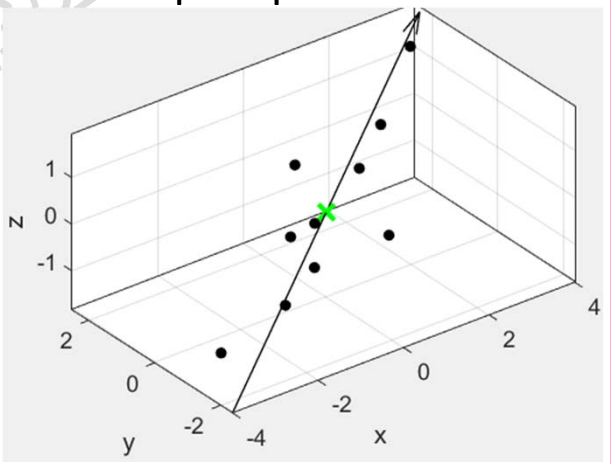
```
[base1,comp1,lambda1] = pca(X,'NumComponents',1);
```

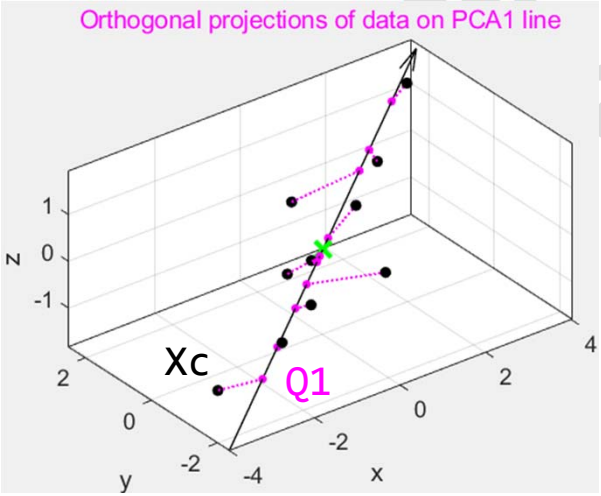
```
[L1,w1] = powerMethod(Xc'*Xc/(N-1));
```

power method: to find the maximum eigenvalue and related eigenvector

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

1st principal direction





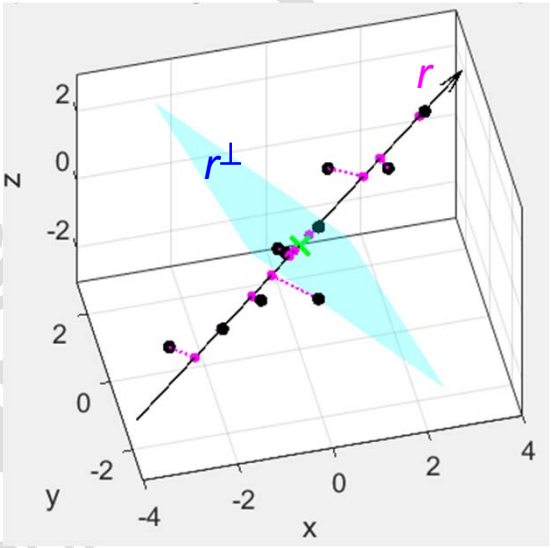

```
lambda1(1)
       4.7561
base1 =
       0.80209
       0.47385
        0.3635
```

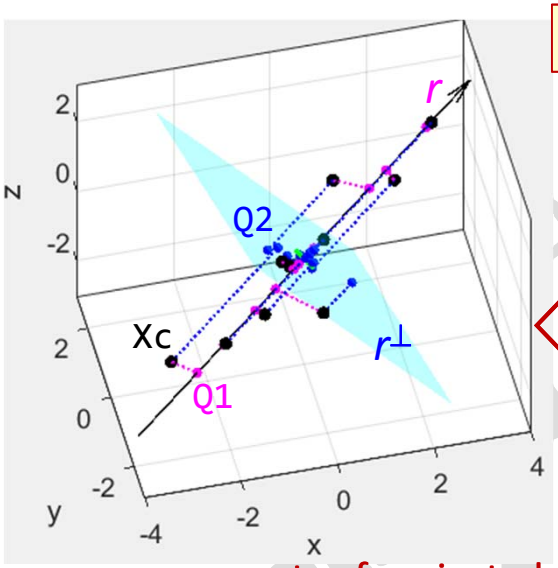
```
L1 =
       4.7561
w1 =
       0.80209
       0.47385
        0.3635
```

```
Pperp=base1*base1';  % orthogonal projection matrix onto line r
Q1=Pperp*Xc'; Q1=Q1';  % comp1 == Q1*base1
plot3(Q1(:,1),Q1(:,2),Q1(:,3),'m.','MarkerSize',18)
title('Orthogonal projections of data on PC1')
```

Incremental PCA algorithm illustration (2)

```
B2=null(base1'); % basis of the orthogonal complement of r
syms a b real;  plane=B2*[a;b];
h=ezsurf(plane(1),plane(2),plane(3),[-3 3]);
title('orthogonal complement of the 1^{st} principal direction')
```



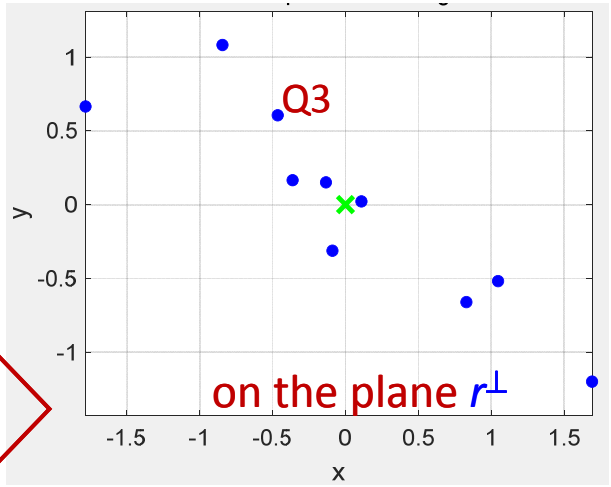


complementary projection matrix I–P

```
Pcmpl=eye(3) - Pperp;
Q2=Pcmpl*Xc'; Q2=Q2'; % 3D projections on r⊥
plot3(Q2(:,1),Q2(:,2),Q2(:,3),'b.','MarkerSize',18)
title('Projections on the orthogonal complement of r')
```

components of projected vectors w.r.t. the plane basis

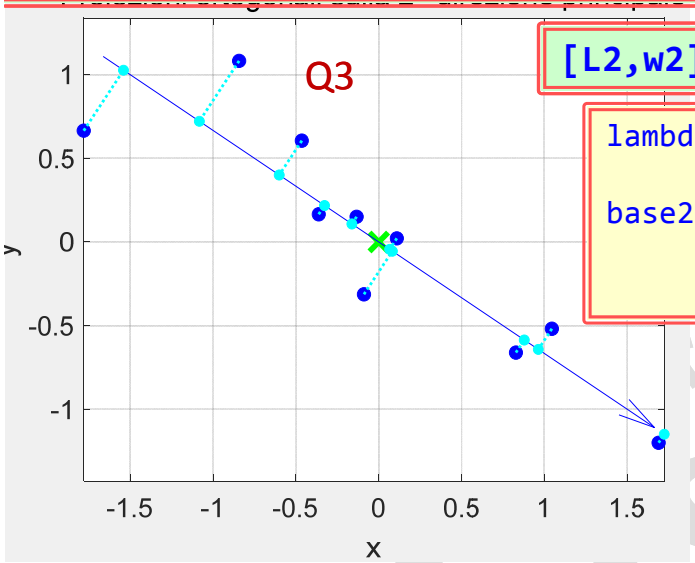
```
Q3=B2'*Q2'; % 2D components w.r.t. B2 (equivalent to  Q3=B2\Q2';)
Q3=Q3';
plot(Q3(:,1),Q3(:,2),'b.','MarkerSize',24); hold on
h=plot(0,0,'Xg'); set(h,'LineWidth',3,'MarkerSize',14)
axis equal; grid on; box on
set(gca,'FontSize',14); xlabel('x'); ylabel('y')
title('Projections on the orthogonal complement of r')
```



Incremental PCA algorithm illustration (3)

PCA applied to 2D projected data

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

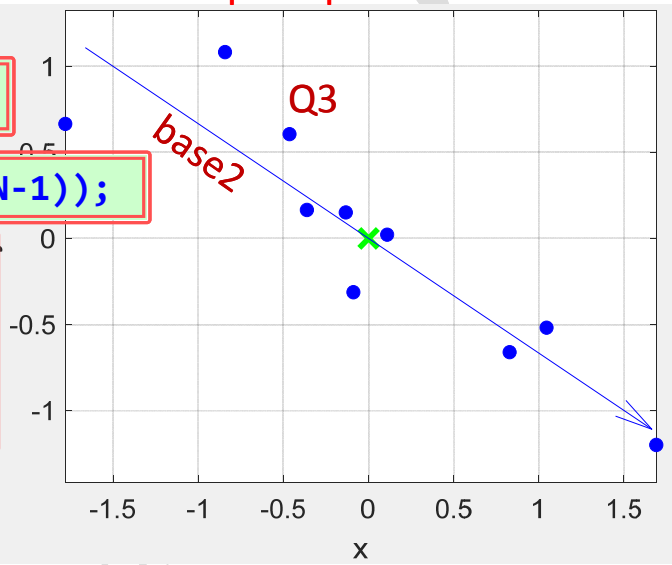



```
[L2,w2]=powerMethod(Q3'*Q3/(N-1));
```

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

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

2nd principal direction



comparison with full PCA

```
[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
```

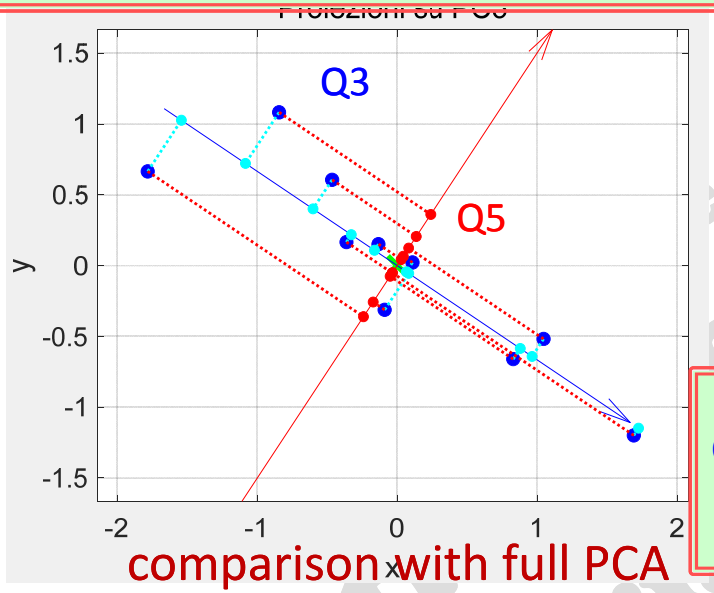
```
comp2 =
     -1.8518
      2.072
      1.158
     -1.3018
    0.079027
    -0.19491
    -0.39336
      1.0563
    -0.72228
    0.098876
```

```
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
```

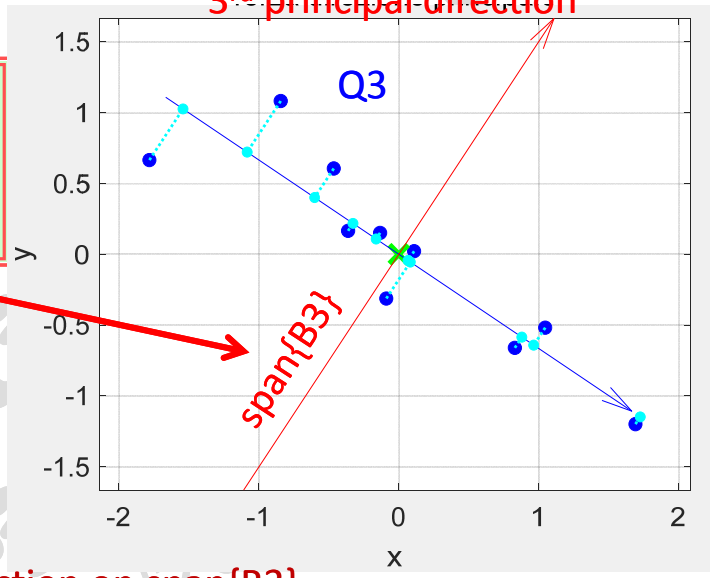
Incremental PCA algorithm illustration (4)

orthogonal complement of base2 on the plane $r^{\perp}$

```
B3=null(base2');  % basis of the orthogonal complement of r
figure(2)
h=quiver(-2*B3(1),-2*B3(2),4*B3(1),4*B3(2),1);
set(h,'Color','r')
```



3rd principal direction



orthogonal projection on 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);
```

comparison with full PCA

```
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
        0.2888       -1.3018        0.43342
       -0.3064      0.079027       0.078779
       -2.4193       -0.1949       0.051441
       -1.4579       -0.3934      -0.063018
      -0.16089        1.0563      -0.090606
        3.6831       -0.7223         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
```

Example PCA: geometrical interpretation

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

basis of $\mathbb{R}^3$

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

features

samples

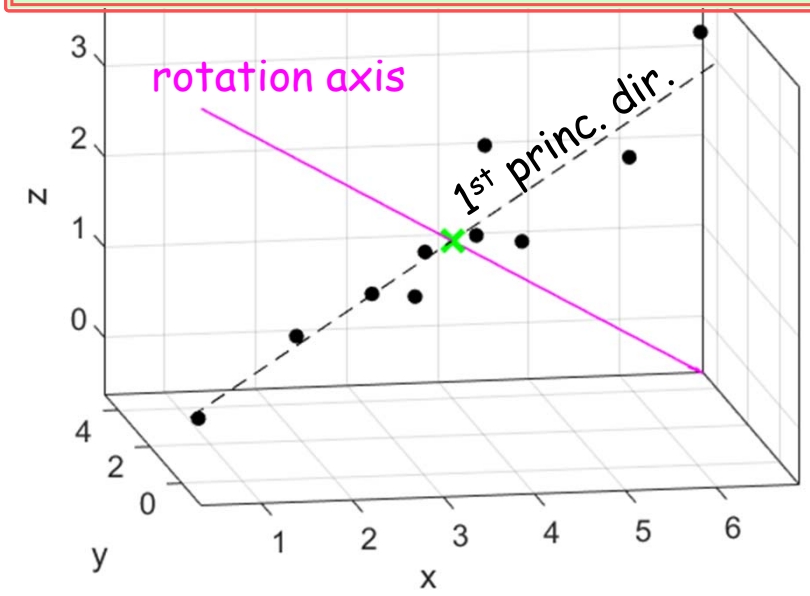
data matrix

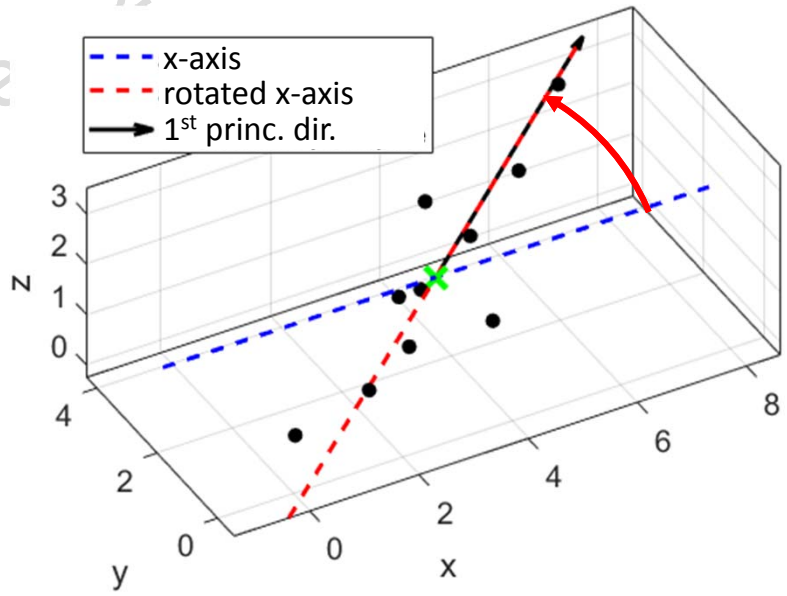
B=basis is an orthogonal matrix.

If det(B)=1 ⇒ it is a 3D rotation around the sample mean

axis and angle of 3D rotation

```
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
```

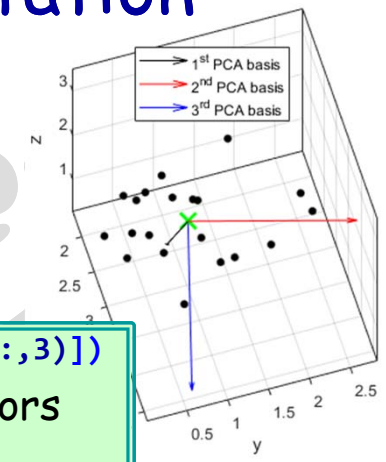




Example PCA: geometrical interpretation

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

it is an improper rotation




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

equal vectors

equal vectors

equal vectors

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

opposite vectors

In order to get a 3D rotation matrix, we can, for instance, change the sign to a single column of **basis***

* Of course, we will also have to change the sign to the corresponding column of **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
```

now it is a proper rotation

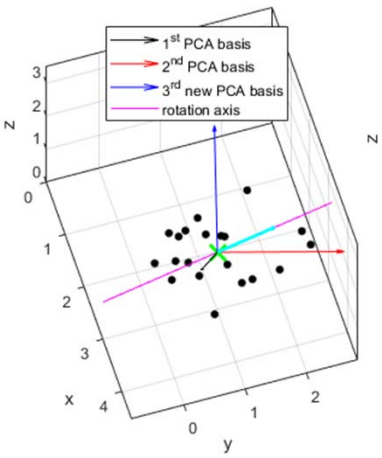
cross(basis(:,1),basis(:,2)) computes a basis for the orthogonal complement of basis(:,1:2)

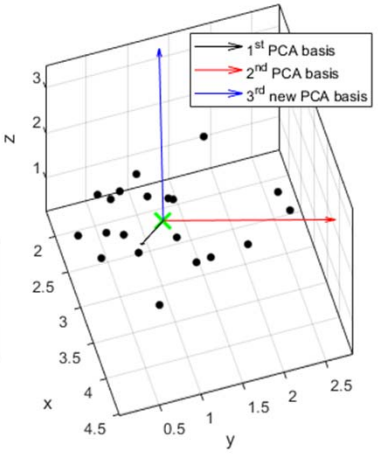
```
disp([cross(basis(:,1),basis(:,2)) null(basis(:,1:2)')])
   -0.74938     -0.74938
    0.20547      0.20547
    0.62945      0.62945
```

```
ax=null(basis - eye(size(basis,1)))
ax =
      0.05614
      0.91699
      0.39494
```

axis and rotation angle

```
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
```





Derivation of the Full PCA algorithm

In the Linear Space $\mathbb{R}^d$, if we have d linearly independent vectors $\{\mathbf{u}_i\}$, they can form a basis of $\mathbb{R}^d$; then $\forall \mathbf{x} \in \mathbb{R}^d$

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

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

components of $\mathbf{x}$ w.r.t. $\mathbf{u}_i$

The same holds for the N samples in the dataset $\mathcal{D} = \{\mathbf{x}^{[n]}\}_{n=1,\ldots,N}$.

Let $\mathbf{x}'$ be the projection of $\mathbf{x}$ with all the d components z_i w.r.t. the new basis $\{\mathbf{u}_1,\ldots,\mathbf{u}_d\}$, and let $\mathbf{x}''$ be the projection with only the first $K < d$ components:

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

$\mathbf{x}''$ is an approximation of $\mathbf{x}'$ in the subspace spanned by the first K vectors $\mathbf{u}_i$

In order to find them later, let us vary the last $d-K$ scalars (b_i) of the linear combination of $\mathbf{x}'$

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

The error in the $\mathbf{x}^{[n]}$ vector is given by:

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

We look for the basis vectors $\mathbf{u}_i$ and the coefficients b_i such that the error in all the dataset be minimum; i.e. we want to minimize the following objective function:

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

minimize SSE (sum of squared errors)

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

The function J (sum of norms, i.e. of convex functions) has a non-trivial minimum ($\neq 0$) only if we add some constraints. We can require that the basis vectors $\{\mathbf{u}_i\}$ be orthonormal: $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{ij}$, where δ_{ij} is the Kronecker symbol ($\delta_{ij}=1$ if $i=j$, $\delta_{ij}=0$ if $i\neq j$). These constraints are written as follows

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

The constrained optimization problem, with only equality constraints, can be solved by the **method of Lagrange multipliers**, that produces an unconstrained minimization problem for the Lagrangian Function $\mathbb{L}$:

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

multipliers

The solution to the previous problem can be found at stationary points (the zeroes of the gradient of $\mathbb{L}$):

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

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

$$\frac{\partial \mathbb{L}}{\partial b_i} = 0 \;\Leftrightarrow\; b_i = \frac{1}{N}\sum_{n=1}^{N}z_i^{[n]} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_i^{\mathsf{T}}\mathbf{x}^{[n]} = \mathbf{u}_i^{\mathsf{T}}\bar{\mathbf{x}} \quad \text{where } \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{[n]} \text{ is the sample mean}$$

By putting b_i in $\mathbb{L}$, and swapping the two summations, we get:

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

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

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

By the symmetric property of the std. dot product, the squared dot product can be written as

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

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

outer product of 2 vectors

⇒

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

$\mathbf{u}_i$ doesn't depend on n

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

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

(multiple of the sample covariance matrix)

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

We have to prove that the minimum of J occurs at the eigenvectors of the scatter matrix (the same eigenvectors and scaled eigenvalues of the covariance matrix).

By properties 2 and 3 of the gradient of the std. dot product, we have

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

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

Prop. 3

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

Prop. 2

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

typical matrix equation for eigenvalues/eigenvectors

scatter matrix

eigenvalues

eigenvectors

⇨

Moreover, the total error (SSE) is given by the sum of the remaining $d-K$ eigenvalues (scaled by 1/2):

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

Full PCA algorithm

If we want **all** the principal components, the incremental PCA algorithm **is not convenient** computationally.

In this case, we:

- Compute efficiently and accurately all the eigenvalues and eigenvectors of the sample covariance matrix (or of the scatter matrix), or alternatively of the correlation matrix.
- Sort the eigenvalues in descent order (and consequently also the related eigenvectors).

To get a dimensionality reduction of the data space, we:

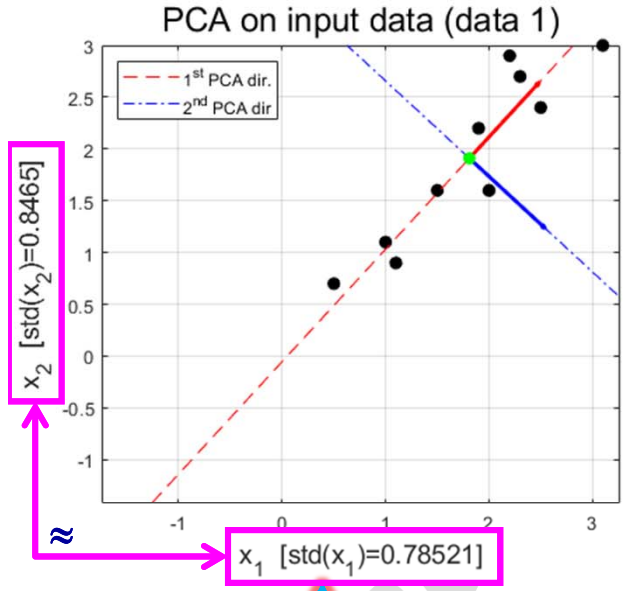
- Compute a normalized cumulative sum of the eigenvalues.
- Compute the minimum index N such that the normalized cumulative sum is not less than a percent tolerance.
- Select the first N eigenvectors as a basis of the "reduced" PCA subspace.
- Compute the components of data w.r.t. this "reduced" PCA basis.

Covariance, scatter, or correlation matrix:

Example 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);
```

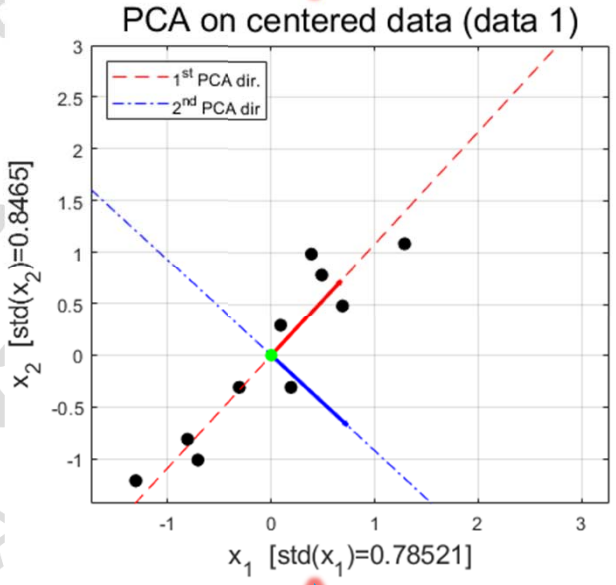
PCA directions are almost equal



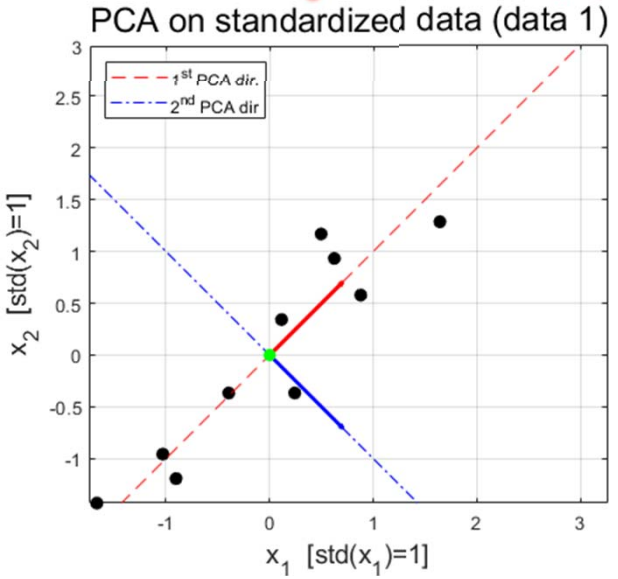
≈

samples with a similar std. deviation

data are scaled, and PCA directions only differ by ~ 2 degrees




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



samples and PCA basis are shifted from μ to $\mathbf{O}$

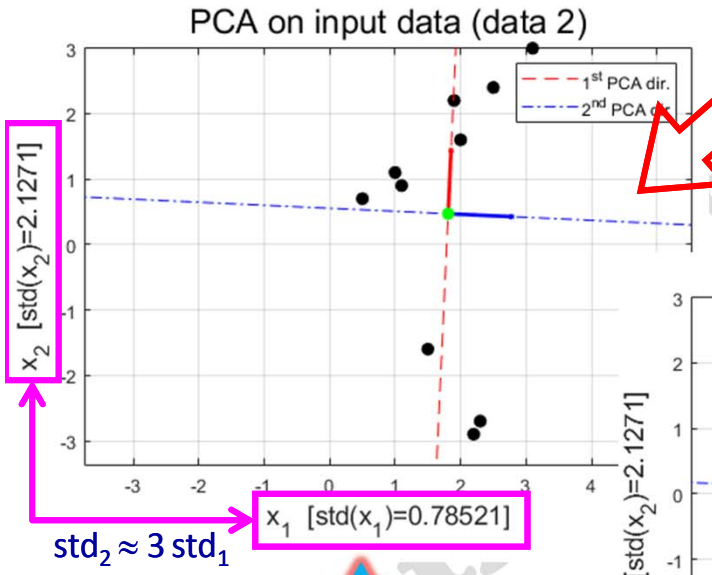
Covariance, scatter, or correlation matrix:

Example 2


```
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);
```

x2: max variance

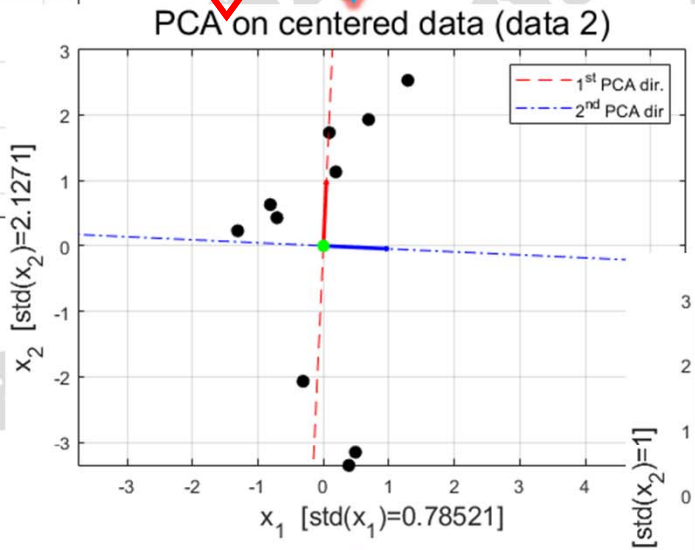
1st PCA direction is almost vertical



std$_2 \approx 3$ std$_1$

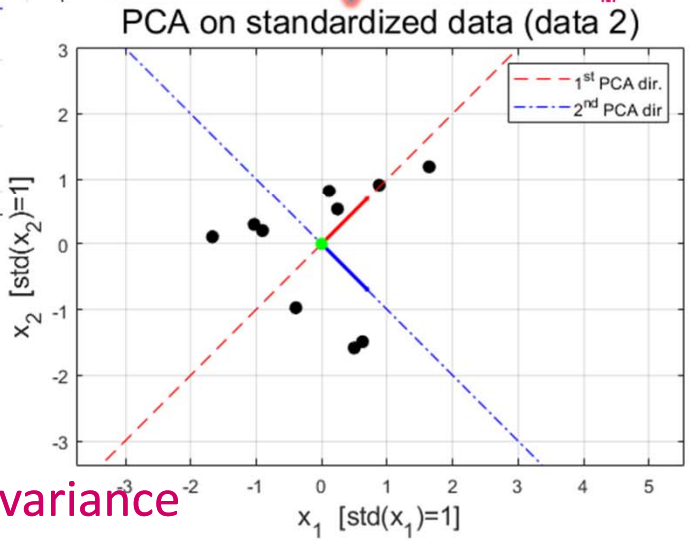
samples with a different std. dev.

data are scaled, and PCA directions differ



the same PCA basis as example 1

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



samples and PCA basis are shifted from μ to $\mathbf{O}$

PCA, on samples, favors the random var. with maximum variance

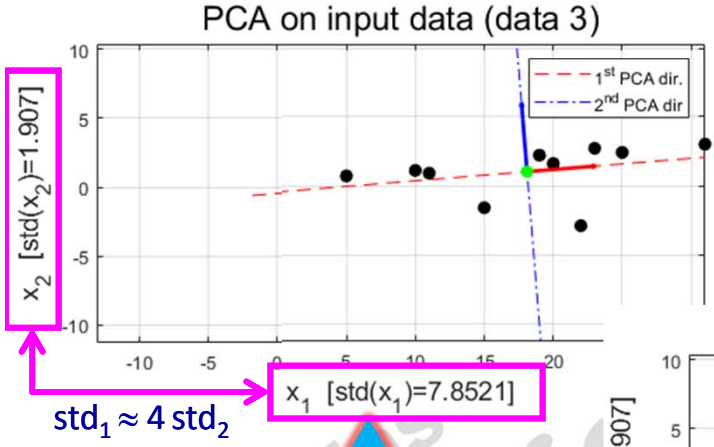
Covariance, scatter, or correlation matrix:

Example 3

10×previous **x1**

```
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);
```

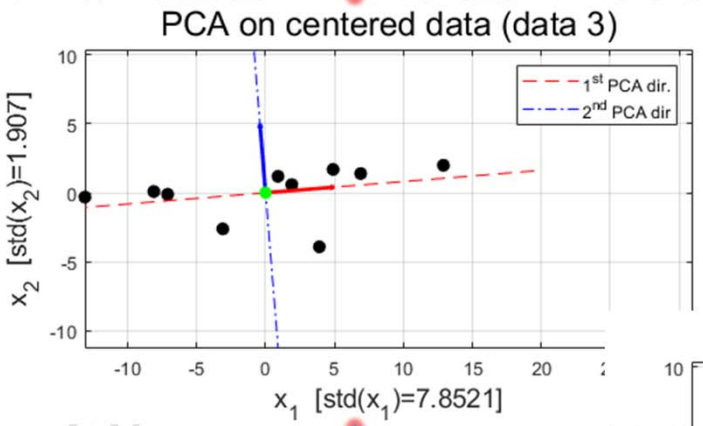
1st PCA direction is almost horizontal



$std_1 \approx 4\, std_2$

campioni con deviazione standard diversa

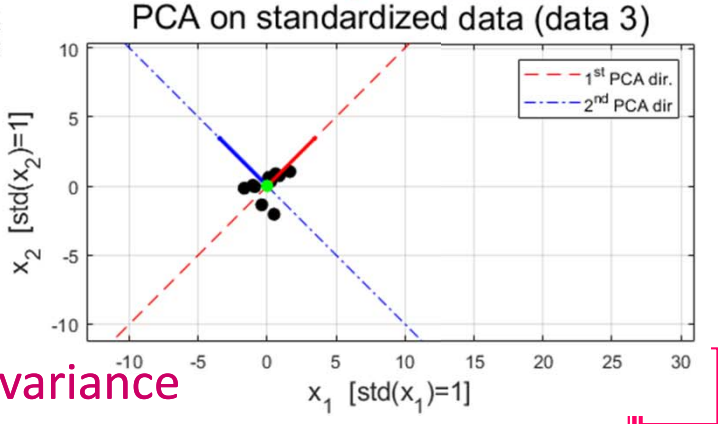
data are scaled, and PCA directions differ



the same PCA basis as example 1

only the 2nd vector has an opposite sign

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



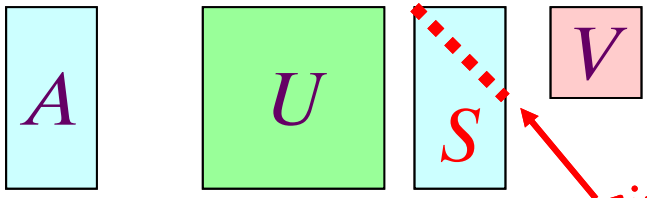
samples and PCA basis are shifted from μ to $\mathbf{O}$

PCA on samples favors the random var. with maximum variance

Compute eigenvalues/eigenvectors by SVD

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

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



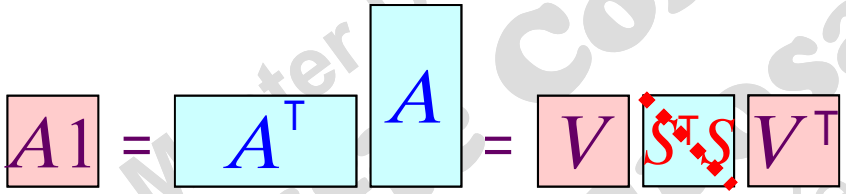
singular values σ_j of A

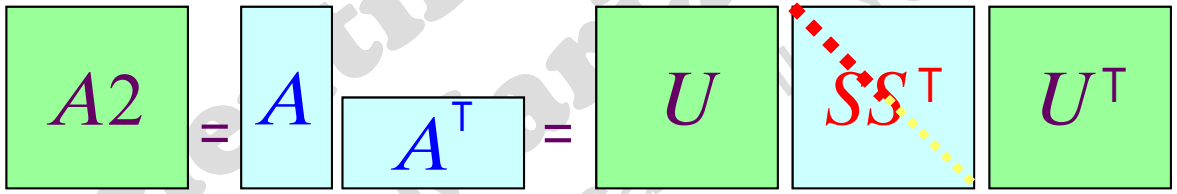
orthonormal columns

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

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

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





eigenvectors

eigenvalues 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)`

Contents

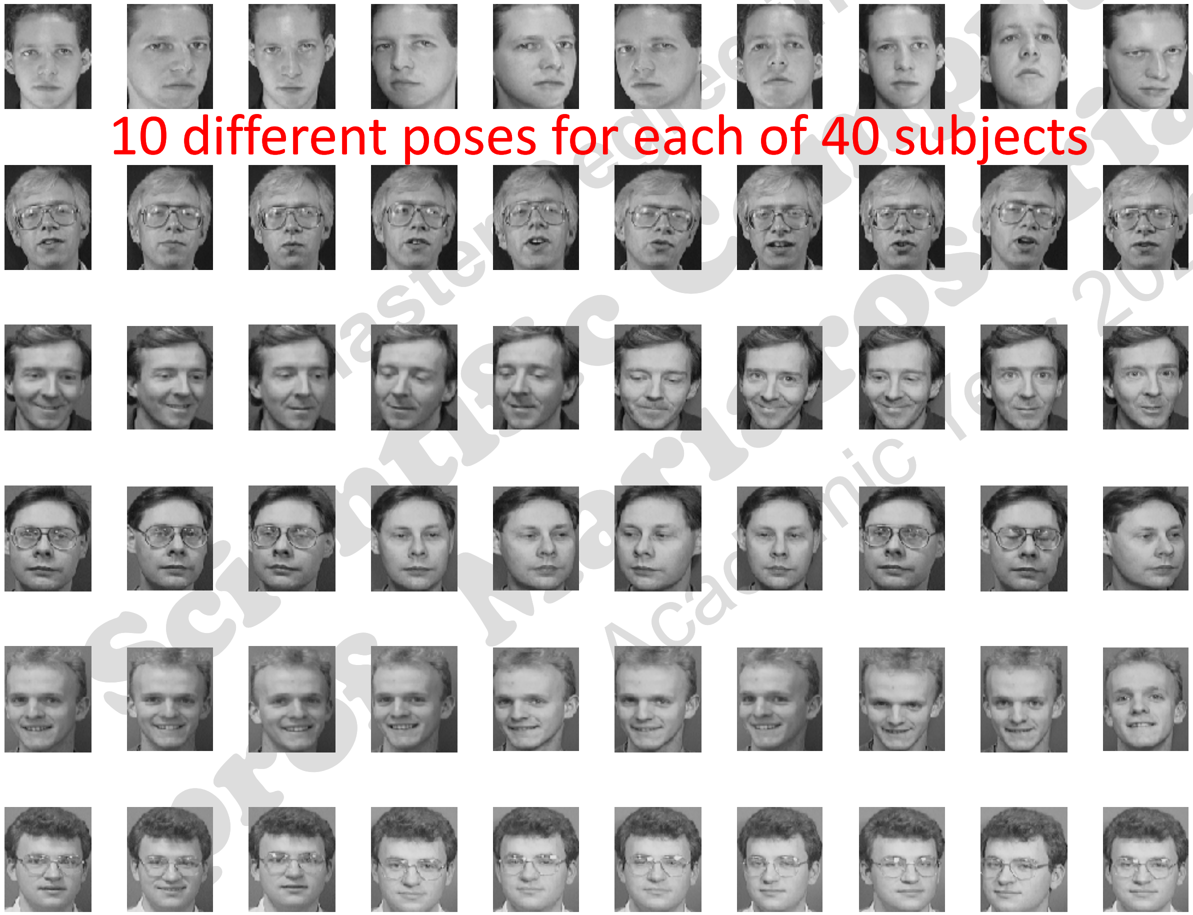
- **PCA application: "Eigenfaces" algorithm.**

"Eigenfaces" Algorithm (PCA)

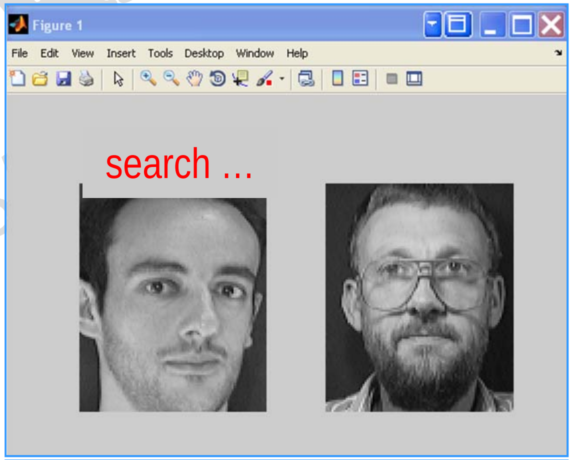
Look for a "face" face database* = 400 images (112×92 pxls) = **4 121 600 bytes**

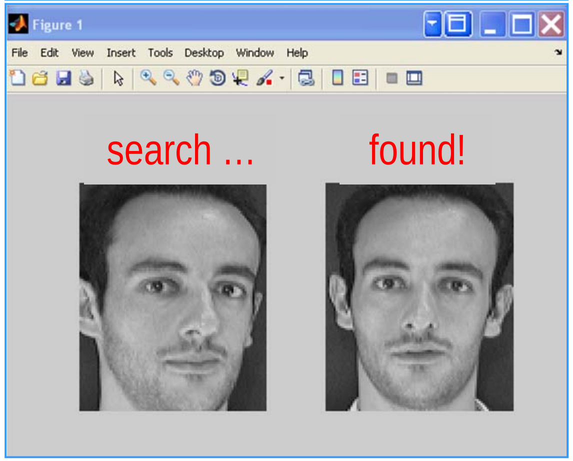
* Download the mat-file: db_400faces_112x92_col_uint8.mat for the face database

- PCA reduces the space to store data (data compression).
- PCA makes it faster to seek a face in the database.



10 different poses for each of 40 subjects





Idea

The problem is treated from a statistical point of view (PCA): sort random variables by decreasing variances.

Is it possible to extract a "reduced basis", from the population of all the "faces", able to maintain data information within a preset tolerance?

Image compression: instead of storing the images, we only save their "*descriptors*" (…), whose number is much less than the number of image pixels.

Faster face search: instead of comparing image pixels (useless!), we only compare their "descriptors" (…), whose number is much less than the total number of pixels.

"Eigenfaces" Algorithm: how to organize the database

The image n. k (112×92 pxls) is saved in a (col-wise) vector X_k, of 10304=112×92 components, and all these columns form a matrix W of **uint8** (8-bit unsigned int).

* Download the mat-file: db_400faces_112x92_col_uint8.mat for the face database

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

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

image #346

X(10304 ×400) is a matrix of double … for floating-point computations

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

to display images

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

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

to display a matrix as an image

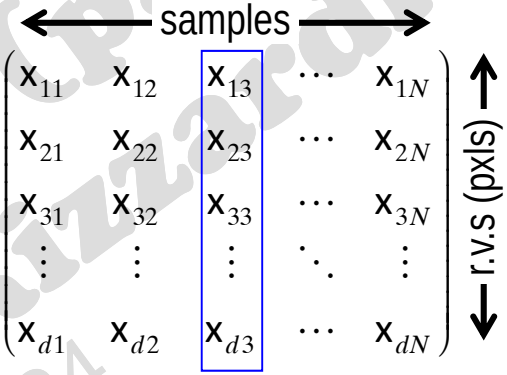
```
imagesc(reshape(X(:,346),m,n))
axis equal; axis off
colormap('gray')
```

MATLAB example: "Eigenfaces" Algorithm

```
load db_400faces_112x92_col_uint8
```

W(10304,400): images are `uint8`

1) Load in the MATLAB Workspace the face numeric database: the data matrix **W**($d \times N$) contains a sample in every column. The *feature space* is d-dimensional. The face complete database contains N=400 images of size 112×92 pxls. *Feature space's* dimension is: d=10304.



1 sample = 1 image

searched subject

#380

```
ri = 380;
```
constant or random
```
ri = randi(size(X,2));
```

```
X = double(W(:,[1:ri-1 ri+1:end]));
```
double: for fl-p. computations

2) In the full database, set the index **ri** of the "face" to be searched, and remove the corresponding vector from data matrix. The index **ri** can be chosen as a constant or randomly.

```
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')
```

mean face

3) Compute the vector μ (sample mean) and the centered data matrix $\mathbf{X}_c$.

If the face to be searched is #380, in the database, the same subject appears in 9 other different poses

4) Apply PCA:

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

the same results as

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

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

PCA *changes the basis* of the space of *centered data vectors* and produces a new basis B of $\mathbb{R}^d$, B=**basis'**, that is an *orthonormal basis* ($BB^T = B^TB = I$):

' because pca requires **Xc'** instead of **Xc**

projection **PCA** : $\mathbf{v} \in Xc \rightarrow \mathbf{w} = B^T\mathbf{v} \in$ PCA *subspace*

reconstruction $\mathbf{PCA}^{-1}$: $\mathbf{w} \in$ PCA *subspace* $\rightarrow \mathbf{v} = B\,\mathbf{w} \in Xc$

w = comp'

Don't forget that we are working with vectors in a Linear Space

projection of Xc onto PCA subspace

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

comp': components w.r.t. PCA basis

reconstruction of Xc from PCA subspace

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

orthonormal basis of $\mathbb{R}^{10304}$

```
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
```

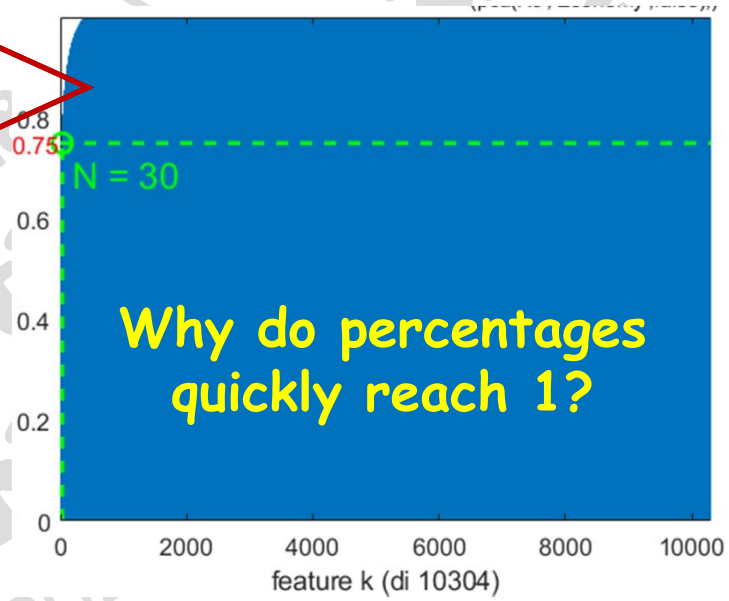
PCA can be numerically computed by means of *eigenvalues* and *eigenvectors* of the covariance matrix or by SVD factorization of X_c.

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

5) In order to set a suitable size for the reduced PCA basis, compute the normalized cumulative sum of the "explained variance" of *features* (*r.v.s*):

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

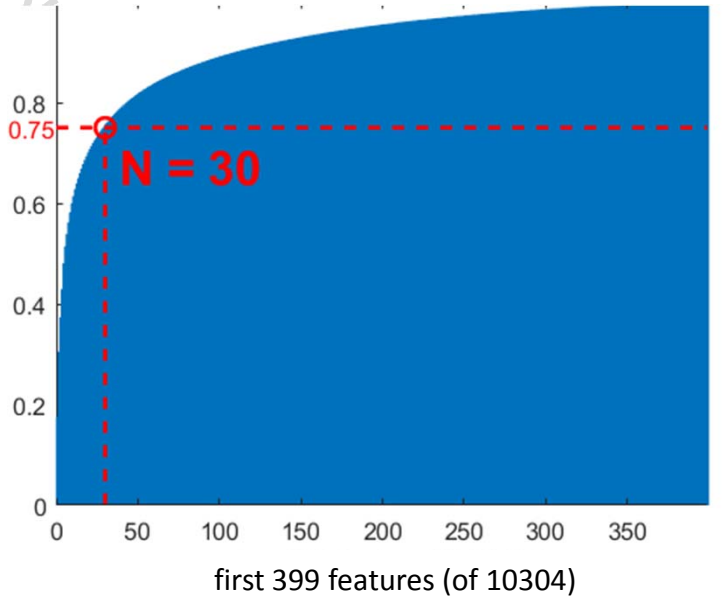
reduced PCA basis



This algorithm reduced the feature space basis from 10304 to 30 vectors.

The first 8 Eigenfaces of a … "spectral basis" (PCA basis)

"spectral" as per eigenvalues, … but also as ghosts



projection onto PCA subspace

```
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) Compute the reduced database, i.e. all the face components w.r.t. the reduced PCA basis.

The face database, with respect to the standard basis, has dimensions **10304×399**.

Now, with respect to the reduced PCA basis, its dimensions are **30×399** (data compression).

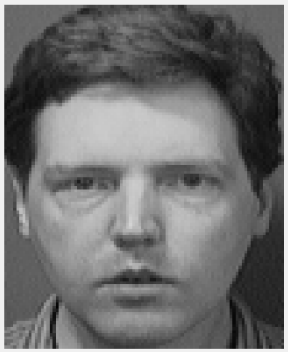
```
Wb=Rbasis'*(double(W(:,ri)) - mu);
```

searched image

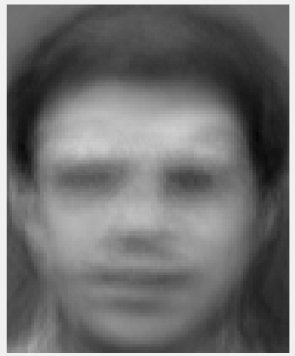
mathematically equivalent to

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

7) Also the searched image is projected onto PCA subspace.

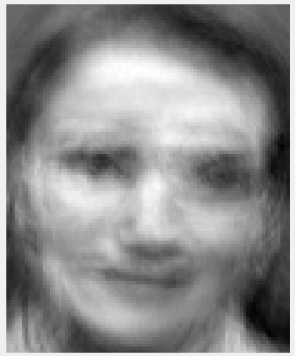


original
searched face



reconstruction of
searched face

face #346



reconstruction of
face#346

8) Search the wanted face in the new reduced database.

To find the subject, two different metrics are used (just to compare their results)

```
DIST = zeros(Nc,1);     SIML = zeros(Nc,1);
for j=1:Nc
    % distance metric: euclidean norm of the residual vector
    DIST(j) = norm(Rcomp(:,j) - Wb);
    % similarity metric: cosine of the angle between the vectors
    SIML(j) = dot(Rcomp(:,j),Wb)/(norm(Rcomp(:,j))*norm(Wb));
end
```

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

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

scalar form

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

matrix form

```
% the best results
Jdist = find(DIST == min(DIST),1,'first');
Jsiml = find(SIML == max(SIML),1,'first');
```

```
% correct the image index back to the whole database
if Jdist >= ri
    JJdist=Jdist+1;
else
    JJdist=Jdist;
end
if Jsiml >= ri
    JJsiml=Jsiml+1;
else
    JJsiml=Jsiml;
end
```

only to display

searched face	closest face	most similar face
		
#380	#375	#375

9) Search the top 4 results:

```
PERCtol=0.75; % percent tolerance
```

```
[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
```

correct the image index back to the whole database

searched subject: #380

1st closest face #375

dist/max(dist)=0.11523
coseno = 0.95971

2nd closest face #372

dist/max(dist)=0.14928
coseno = 0.93431

3rd closest face #377

dist/max(dist)=0.20234
coseno = 0.86906

4th closest face #373

dist/max(dist)=0.24896
coseno = 0.82538

1st most similar face #375

dist/max(dist)=0.11523
coseno = 0.95971

2nd most similar face #372

dist/max(dist)=0.14928
coseno = 0.93431

3rd most similar face #377

dist/max(dist)=0.20234
coseno = 0.86906

4th most similar face #376

dist/max(dist)=0.27
coseno = 0.82578

PERCtol=0.75; % percent tolerance

1st closest face #156

dist/max(dist)=0.65734
coseno = 0.055058

2nd closest face #152

dist/max(dist)=0.655
coseno = 0.063914

3rd closest face #153

dist/max(dist)=0.65608
coseno = 0.060561

4th closest face #146

dist/max(dist)=0.65049
coseno = 0.10322

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

another search

#157

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

1st most similar face #156

dist/max(dist)=0.65734
coseno = 0.81892

2nd most similar face #152

dist/max(dist)=0.655
coseno = 0.65951

3rd most similar face #155

dist/max(dist)=0.65714
coseno = 0.58542

4th most similar face #376

dist/max(dist)=0.65608
coseno = 0.56828

Exercise

Implement the "Eigenfaces" algorithm*, and compute eigevalues/eigenvectors by means of MATLAB functions:

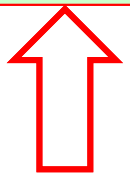
pca(): PCA of X, and of Xc, and of Xs;

svd(): SVD factorization of Xc, and of Xs;

eig(): eigenvalues/eigenvectors of Xc*Xc', and of Xs*Xs';

and compare their results. Moreover compare their execution times with

tic, …, **T=toc**.



put here the code to be evaluated

* Download the mat-file db_400faces_112x92_col_uint8.mat for the face database

see MATLAB Help Browser to use **tic**, …, **T=toc**.

Exercise

Write a MATLAB function to implement the "Incremetal PCA" algorithm (equipped by a "power method" function), and compare its results with those returned by the MATLAB function **`pca()`**.

Compare also their execution times by means of

`tic`, …, **`T=toc`**.

Contents

- Regression in 2D and in 3D.
- PCA vs Least Squares.

Regression line in $\mathbb{R}^2$

In Statistics the "Linear Least Squares" line is said the **regression line**.

Unlike **correlation**, which indicates some stochastic dependence between the random variables x and y, **regression** denotes some functional dependence between x and y:

- $y=\Phi(x)$, i.e. y depends on x,
- or $x=\Psi(y)$, i.e. x depends on y.

In particular, for **linear regression** $\Phi(x)$, or $\Psi(y)$, is a linear function.

- Linear Regression of y on x: $y=\Phi(x)=ax+b$

  The following functional has to be minimized

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

- Linear Regression of x on y: $x=\Psi(y)=cy+d$

  The following functional has to be minimized

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

Regression line of y on x in $\mathbb{R}^2$

By definition the "Least Squares" line r : $y=ax+b$, w.r.t. the data $(x_i,y_i)_{i=1,\ldots,n}$, minimizes the the sum of squared residuals: min $J_{LS}(a,b)$

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

$\nabla J_{LS}(a,b) = 0$

$$\begin{cases}\dfrac{\partial}{\partial a}J_{LS}^{(1)}=0 & \Leftrightarrow \quad 2\displaystyle\sum_{i=1}^{n}\left[y_i-(ax_i+b)\right]x_i=0\\[2ex] \dfrac{\partial}{\partial b}J_{LS}^{(1)}=0 & \Leftrightarrow \quad 2\displaystyle\sum_{i=1}^{n}\left[y_i-(ax_i+b)\right]=0\end{cases}$$

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

$$\Downarrow$$

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

$$\Downarrow$$

mean

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

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

i.e. the Least Squares line, for the given data points $(x_i,y_i)_{i=1,\ldots,n}$, passes through the point sample mean $P_\mu\equiv(\bar{x},\bar{y})=\left(\frac{1}{n}\sum_{i=1}^{n}x_i,\frac{1}{n}\sum_{i=1}^{n}y_i\right)$

Least Squares line in $\mathbb{R}^2$ with MATLAB

$$r : y = ax + b$$

```
%% P: 2D sample points from N(μ,Σ)
N=10;  P=mvnrnd([3 1],[1 .2; .2 .7],N);
B0=mean(P); % sample mean (barycenter)
```

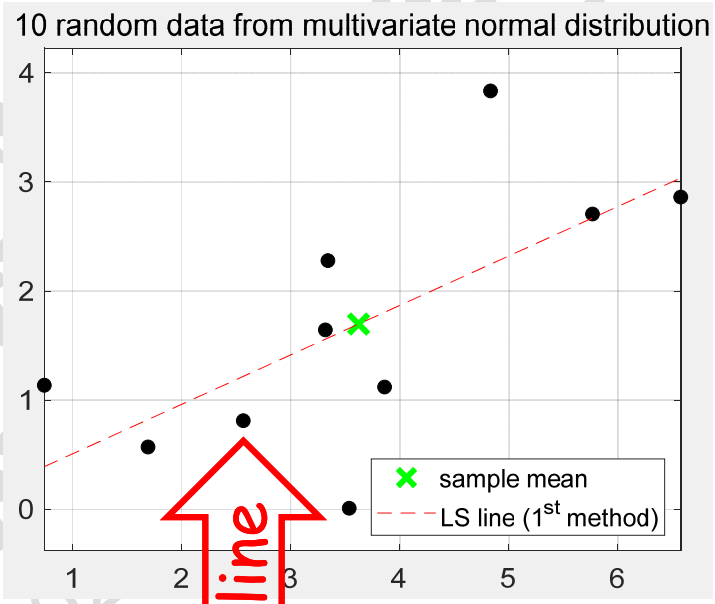
1st method: only for $\mathbb{R}^2$

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

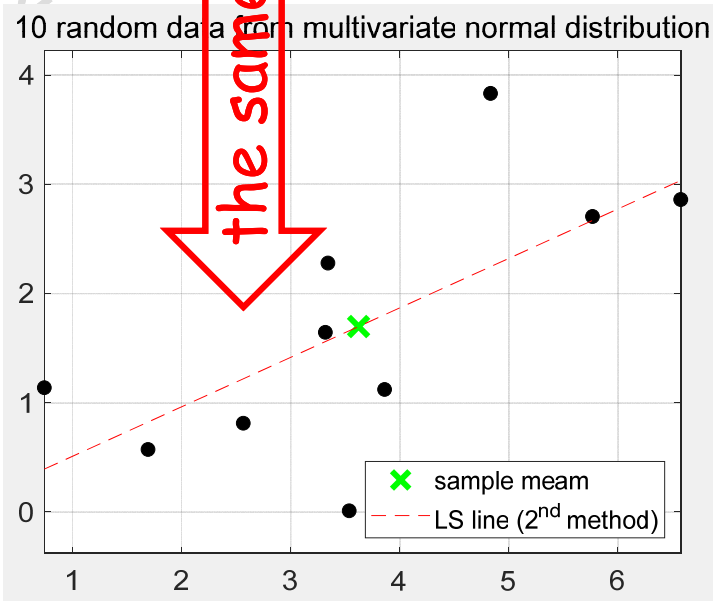
2nd method: overdetermined system
more general

$$y_i = ax_i + b, \quad i = 1,\dots,n \qquad \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

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



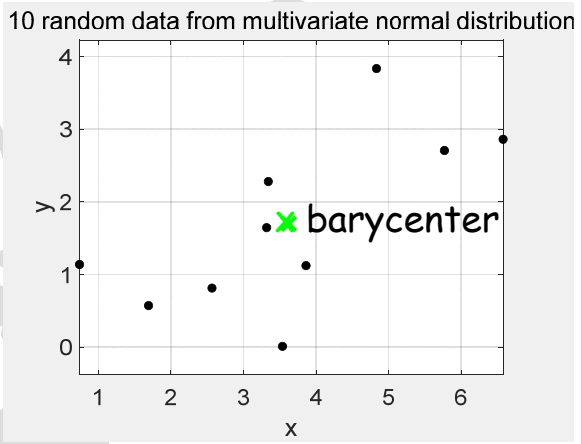
the same line



Regression lines in $\mathbb{R}^2$ with MATLAB

```
%% P: 2D sample points from N(μ,Σ)
N=10; MU=[3 1]; SIGMA=[1 .2; .2 .7];
P=mvnrnd(MU,SIGMA,N);
B0=mean(P); % barycenter (sample mean)
Xi=P(:,1); Yi=P(:,2);
plot(Xi,Yi,'ok','MarkerFaceColor','k')
axis equal; grid on
```

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



regression of y over x

$$r : y = ax + b$$

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

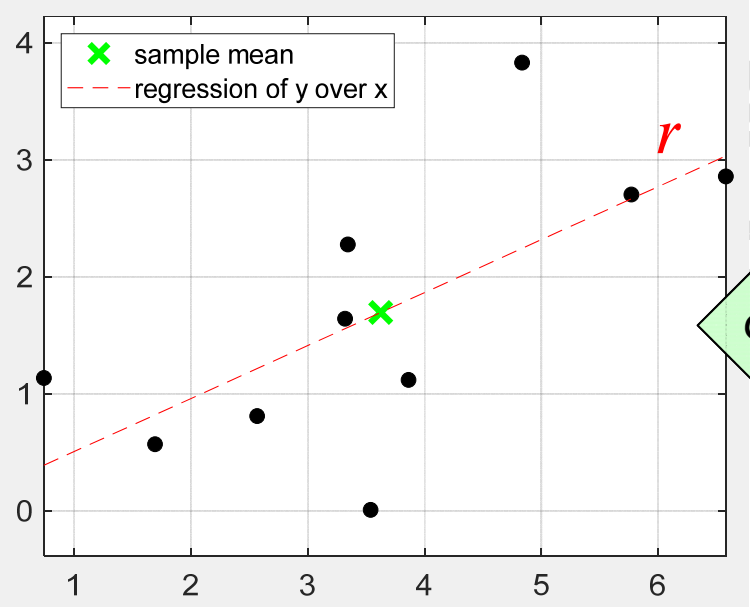
$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

regression of x over y

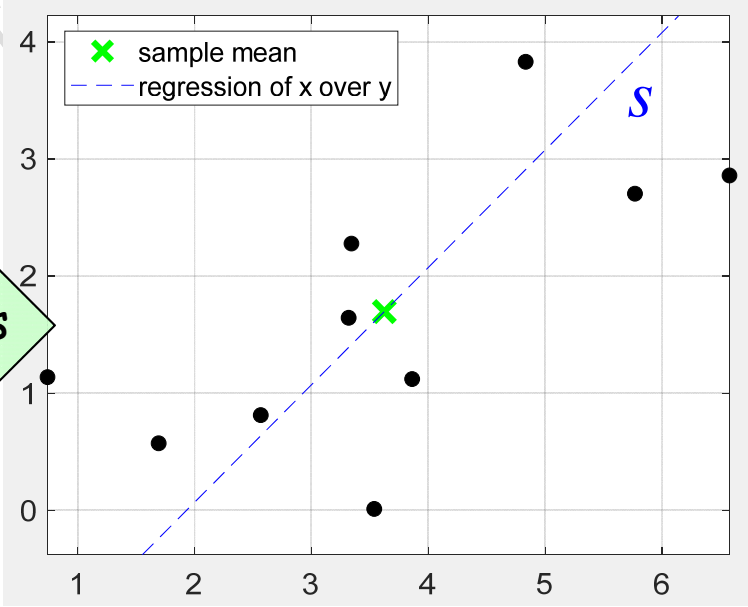
$$s : x = cy + d$$

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

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



different lines



Regression lines in $\mathbb{R}^2$ with MATLAB

regression of y over x

$$r: y = ax + b$$

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

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

regression of x over y

$$s: x = cy + d$$

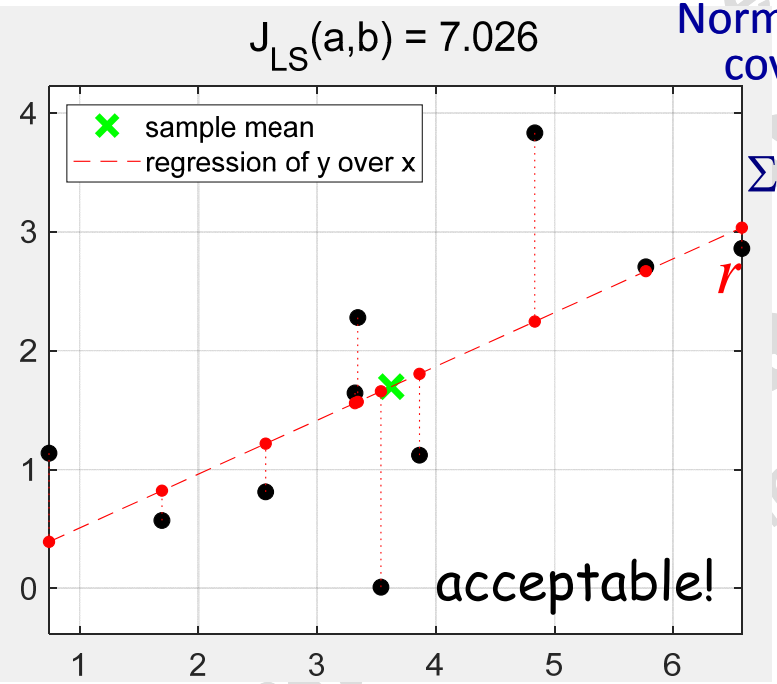
$$x_i = cy_i + d, \quad i = 1,\ldots,n$$

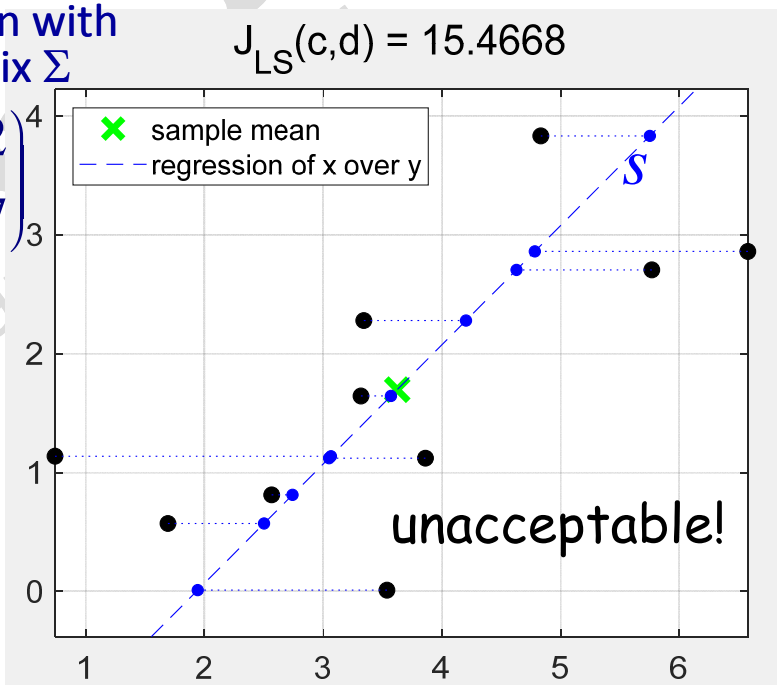
$$J_{LS}^{(2)}(c,d) = \sum_{i=1}^{n}\left[x_i - (cy_i + d)\right]^2$$

The same sample from a Normal distribution with covariance matrix Σ

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

N=10





Exercise: produce similar plots for N=20 with the value of J_{LS}

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

By definition the "Least Squares" plane $\pi : z=ax+by+c$, w.r.t. the data $(x_i,y_i,z_i)_{i=1,\dots,n}$, minimizes the the sum of squared residuals: min $J_{LS}(a,b,c)$

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

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

As already said about the Least Squares line, also the **Least Squares plane**, for the data $(x_i,y_i,z_i)_{i=1,\dots,n}$, passes through the sample mean point

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

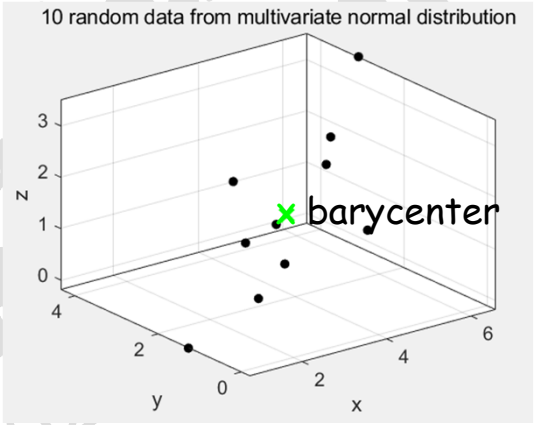
In general the Least Squares hyperplane in $\mathbb{R}^d$, of dimension d–1 and related to the n data ($n>d$) $P_i\left(x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}\right)_{i=1,\dots,n}$ minimizes the functional J_{LS}

Regression plane of z over x,y in $\mathbb{R}^3$ with MATLAB

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

Sample from a Normal distribution with covariance matrix Σ

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



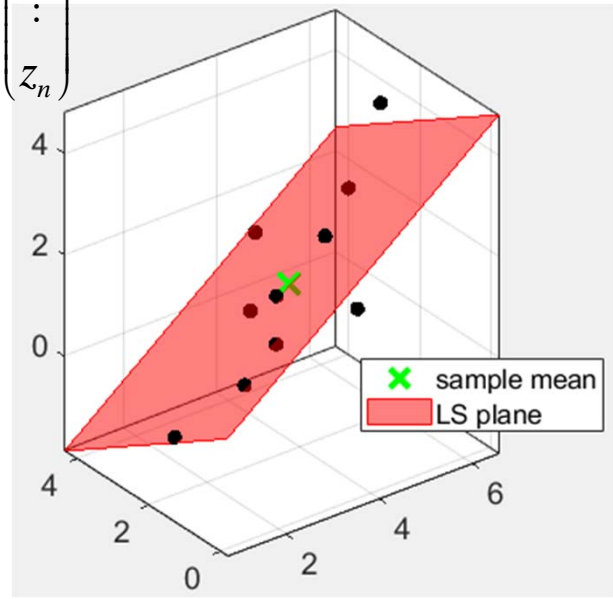

```
%% P: 3d sample points from N(μ,Σ)
N=10; P=mvnrnd([3 1 1],[1 .2 .7; .2 1 0; .7 0 1],N);
B0=mean(P); % sample mean (barycenter)
Xi=P(:,1);  Yi=P(:,2);  Zi=P(:,3);
```

2nd method (more general)

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

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

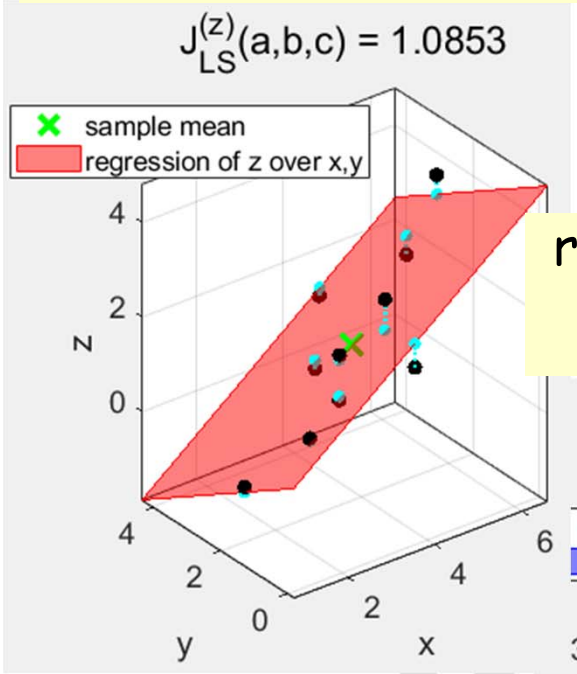
```
%% Least Squares plane
A=[Xi Yi ones(N,1)];
coefZ = A\Zi;
```



Regression planes in $\mathbb{R}^3$ with MATLAB

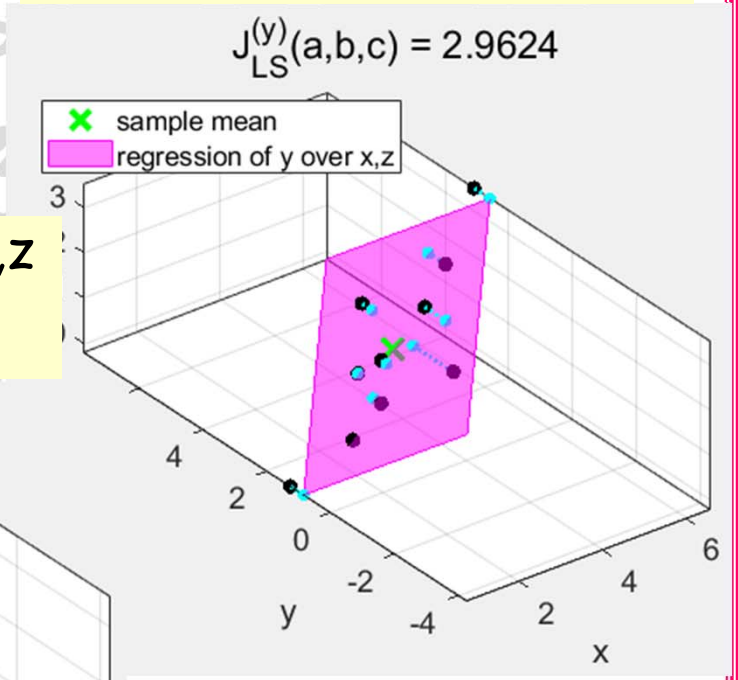
regression of z over x,y

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



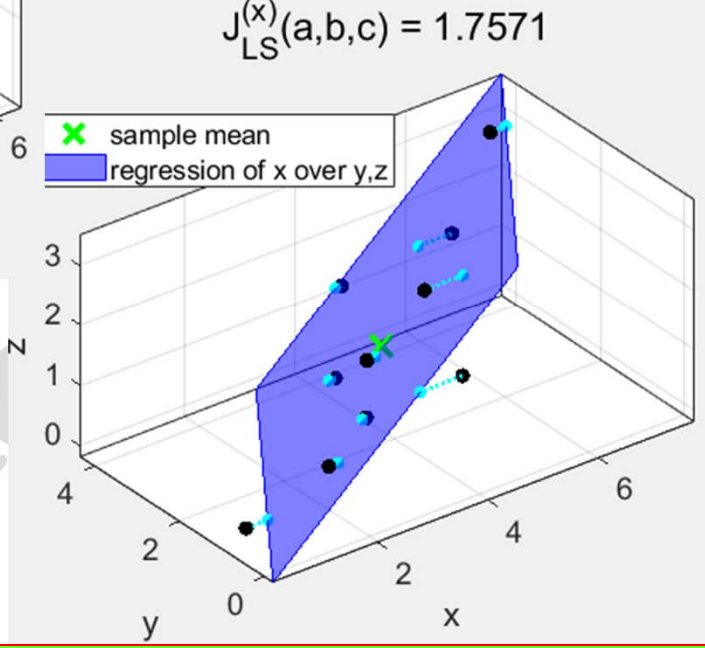
regression of y over x,z

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



regression of x over y,z

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



Exercise: produce similar plots for N=20 with the value of J_{LS}

Least Squares line VS PCA line

Input data: $P_i\left(x_i, y_i\right)_{i=1,\ldots,n}$

sample mean: $P_\mu(\bar{x},\bar{y})$:

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

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

- The Least Squares line minimizes the functional J_{LS}:

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

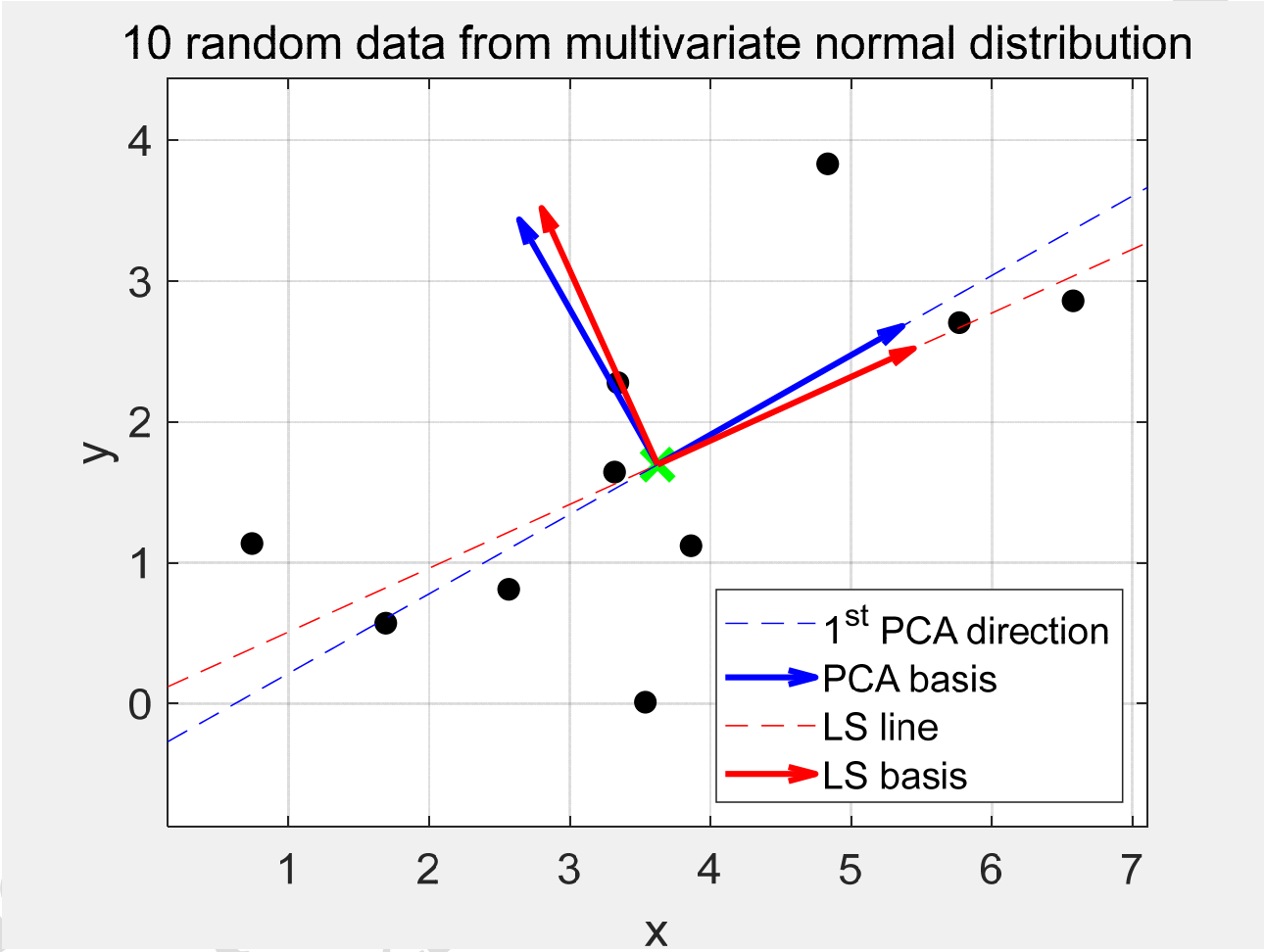
where $A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$

- The 1st principal direction line minimizes the functional J_{PCA}:

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

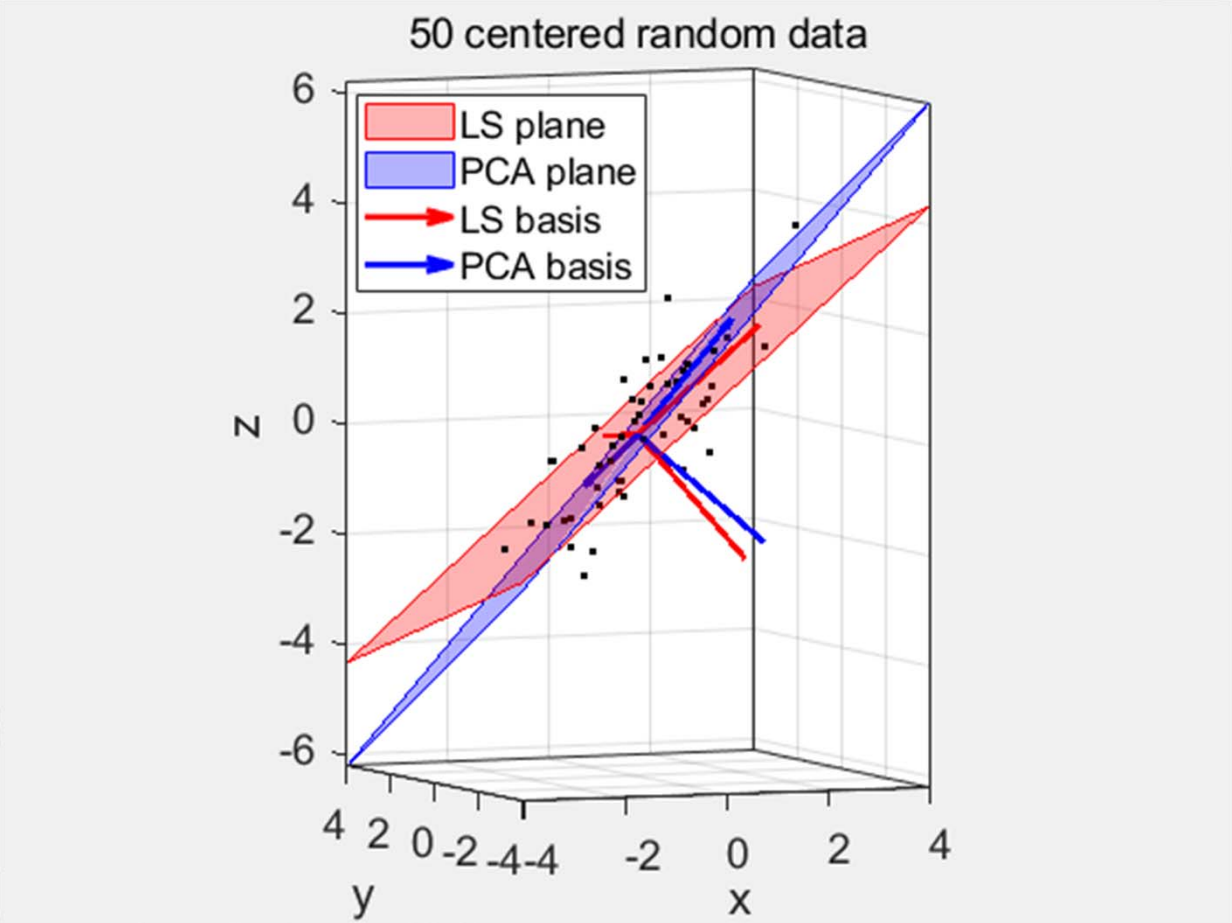
where P_i^{PCA} is the orthogonal projection of P_i on the line passing through P_μ and parallel to **e**, which is the eigenvector related to the maximum eigenvalue of the data covariance matrix.

Least Squares line VS PCA line



Why do the Least Squares line and the PCA line differ?
What optimal condition does each line meet with respect to the samples? How to verify this?

Similarly: Least Squares plane VS PCA plane



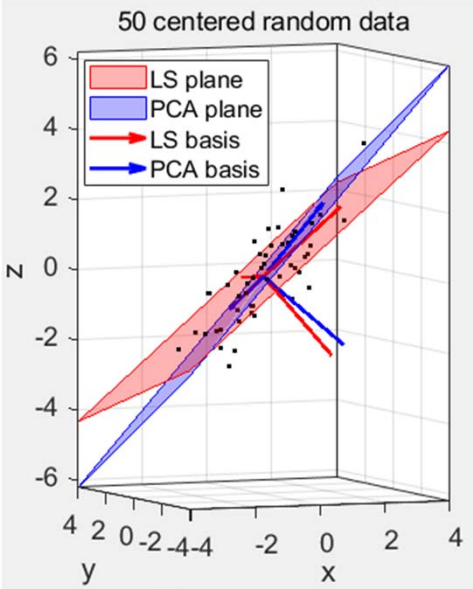
Why do the Least Squares plane and the PCA plane differ?
What optimal condition does each plane meet with respect to the samples? How to verify this?

Least Squares plane VS PCA plane

```
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
[basis, comp, lambda] = pca(P);
plot3(P(:,1),P(:,2),P(:,3),'k.'), axis equal; hold on
% PCA plane: plane passing through B0 and spanned by basis(:,1:2)
%            the plane coefficients are given by basis(:,3)
AX=axis; Px=[AX(1:2);AX(1:2)]; Py=[AX(3:4);AX(3:4)]';
Pz=B0(3)-(basis(1,3)*(Px-B0(1))+basis(2,3)*(Py-B0(2)))/basis(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)
```

or, equivalently

```
% PCA plane: plane passing through B0 and spanned by basis(:,1:2)
syms a b real;  plane=basis(:,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)
```



Exercise

In addition to the PCA plane (first two principal directions), draw the LS plane starting from the same random samples.

What optimal condition does each plane meet with respect to the samples? How to verify this?