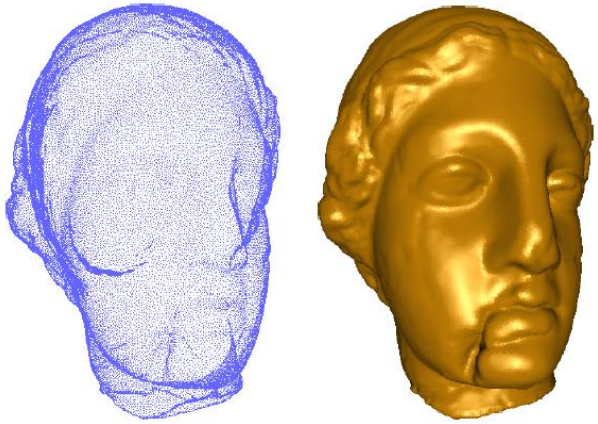
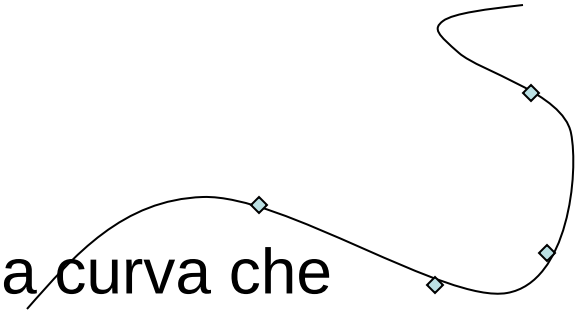
Least Squares Approximation

- Normal Equations
- Method LS-QR
- Method LS-SVD
- PCA
- Local 3D fitting

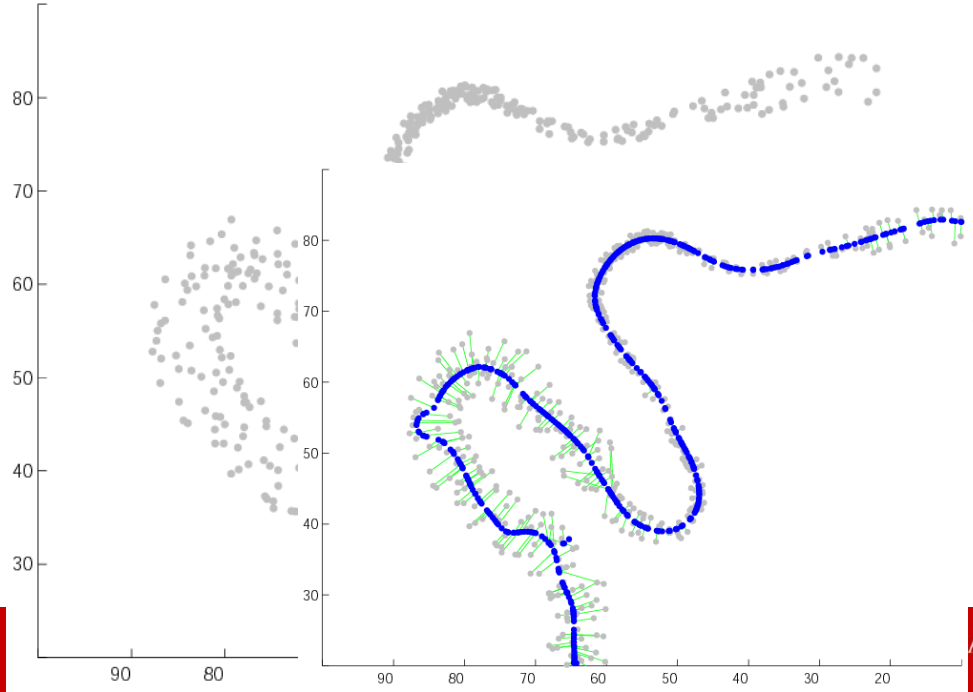


Qual è la differenza tra approssimazione e interpolazione?

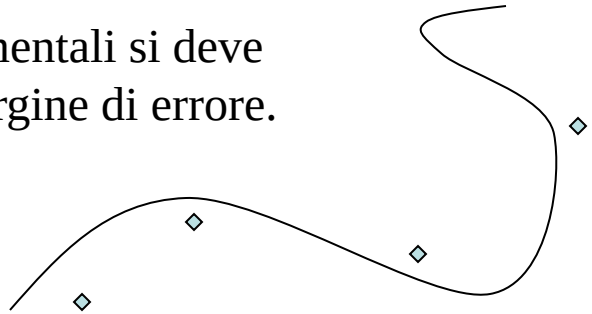
Interpolazione - passa per i punti dati.



Approssimazione – si vuole costruire la curva che approssima i dati minimizzando la **distanza** tra la curva e i dati stessi.



sperimentali si deve
un margine di errore.

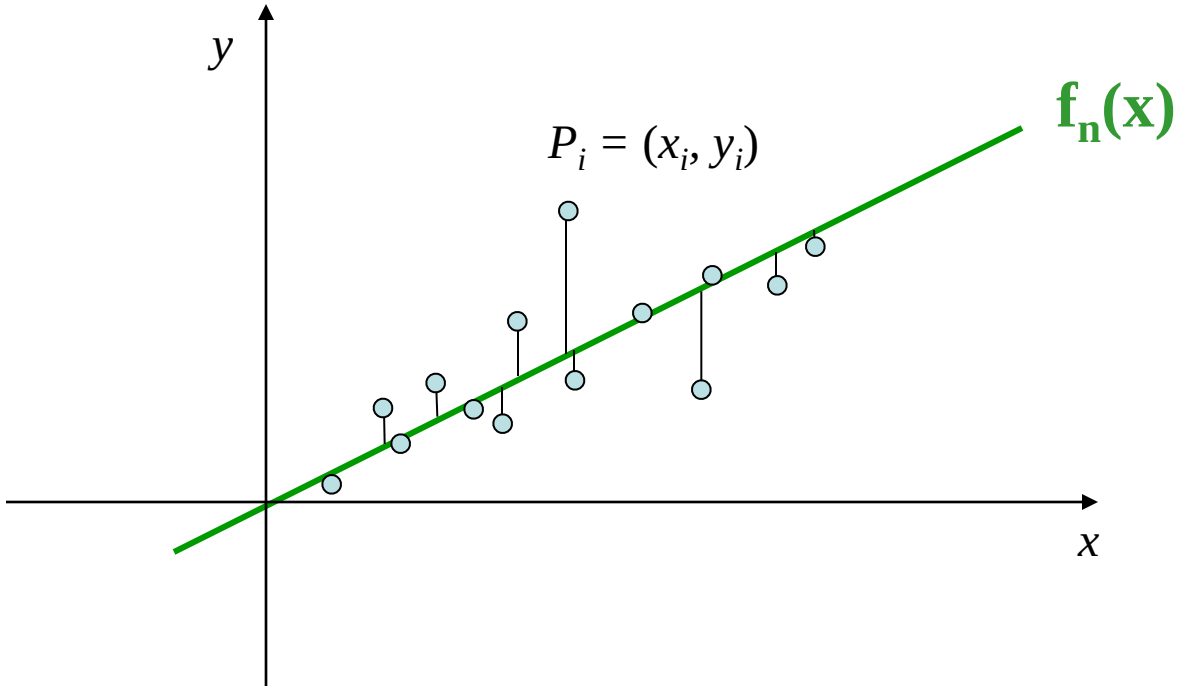


Che tipo di approssimazione?

Scelta della **DISTANZA** con la quale si vuole approssimare i dati (misura dell'errore)

- Pochi dati affidabili ⟶ interpolazione
- Insieme discreto di dati (es. dati sperimentali affetti da variabilità statistica)
  - ⟶ approssimazione ai minimi quadrati
  - ⟶ approssimazione uniforme

Approssimare i punti (x_i,y_i),i=1,..,m,con la funzione $f_n(x)$



minimizzare gli spostamenti in y

Distanza (norma)

- Approssimazione ai minimi quadrati

  $f_n(x)$ vuole minimizzare la quantità:

$$\|y - f_n\|_2 = \left[\sum_{i=1}^{m} (y_i - f_n(x_i))^2 \right]^{\frac{1}{2}},$$

- Approssimazione uniforme

  $f_n(x)$ vuole minimizzare la quantità:

$$\|y - f_n\|_\infty = \max_{i=1,..,m} |y_i - f_n(x_i)|$$

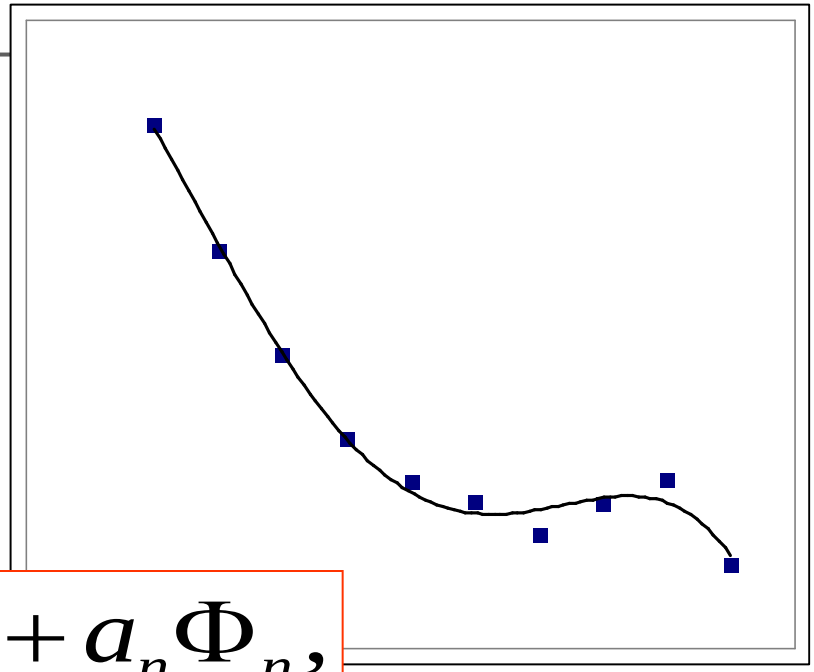
Approssimazione ai minimi quadrati

Assegnati i punti (x_i,y_i), i=1,..,m,

scelte n+1 funzioni base

$$\Phi_0, \Phi_1, .., \Phi_n, \ (n << m)$$

si vuole costruire la funzione di approssimazione



$$f_n(x) = a_0\Phi_0 + a_1\Phi_1 + .. + a_n\Phi_n,$$

individuata con il criterio dei minimi quadrati, ossia i coefficienti $a_{i,\ i=0,..,n}$ sono determinati in modo che lo scarto quadratico medio S sia minimo

$$S = \sum_{i=1}^{m} (y_i - f_n(x_i))^2 = \|y - f_n(x)\|_2^2$$

$\mathbf{y} - \mathbf{f_n}(\mathbf{x})$ è il vettore di elementi

$$y_i - \sum_{j=0}^{n} a_j \phi_j(x_i) \quad i = 1,..,m$$

$$\mathbf{y - Ha} \qquad \mathbf{H}|_{ij} = \phi_j(x_i)$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ \dots \\ \dots \\ y_m \end{bmatrix} - \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \dots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_n(x_1) \\ & & & \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \phi_0(x_m) & \phi_1(x_m) & \dots & \phi_n(x_m) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix}$$

Vettore **a** : parametri incogniti dell'approssimante

METODO DELLE EQUAZIONI NORMALI

$$\min_a S = \|y - Ha\|_2^2 \quad \Leftrightarrow \quad H^T Ha = H^T y$$

Equazioni normali

Verifica

$$S(a) = \| y - Ha \|_2^2 = (y - Ha)^T (y - Ha) =$$
$$= y^T y - y^T Ha - a^T H^T y + a^T H^T Ha =$$
$$= y^T y - 2y^T Ha + a^T H^T Ha$$
$$\nabla S(a) = 0 = -2y^T H + 2H^T Ha = 0$$
$$\Rightarrow \quad H^T Ha = H^T y$$

Sistema simmetrico

METODO DELLE EQUAZIONI NORMALI

$$\min S = \|Ha - y\|_2^2 \quad \Leftrightarrow \quad H^T Ha = H^T y$$

H^TH e' definita positiva se e solo se la matrice H ha rango massimo.

In questo caso, A è non singolare, quindi il problema ha un'unica soluzione.

Attenzione al MALCONDIZIONAMENTO:

$$\mathbf{K(H^TH)=K(H)^2}$$

Questo comporta che errori di arrotondamento nella risoluzione del sistema possono causare grandi errori nell'approssimante dei dati.

Approssimazione ai minimi quadrati (Least Squares - LS)

TEOREMA

Il problema LS ammette sempre soluzione.

La soluzione e' unica se e solo se la matrice H ha rango massimo (le colonne di H sono linearmente indipendenti), se invece le colonne di H sono linearmente dipendenti allora il problema ha infinite soluzioni; tuttavia quella di lunghezza (euclidea) minima e' unica.

Minimi Quadrati Pesati

- Associamo un peso w_i a ciascun punto:

$$\min_{a_0,a_1,\ldots,a_n} \sum_{i=1}^{N} w_i [y_i - f_n(x_i)]^2, \quad w_i > 0 \text{ e non dipendono da } a_i$$

$$\Updownarrow$$

$$\min_{a} (Ha - y)^T W (Ha - y), \quad W_{ii} = w_i \text{ è matrice diagonale } m \times m$$

$$\Updownarrow$$

$$(H^T W H)a = H^T W y \quad \text{sistema equazioni normali nxn}$$

Polinomi ortogonali nelle equazioni normali

Si può evitare la risoluzione del sistema delle equazioni normali scegliendo come funzioni base polinomi ortogonali

$$\left\langle \Phi_k, \Phi_j \right\rangle = \sum_{i=0}^{m} w_i \Phi_k(x_i) \Phi_j(x_i) = 0 \quad k \neq j$$

con matrice dei pesi W e matrice del sistema $H^T W H a = H^T W y$ diagonale di coefficienti:

$$a_j^* = \frac{\sum_{i=0}^{m} w_i f_i \Phi_j(x_i)}{\sum_{i=0}^{m} w_i \Phi^2{}_j(x_i)}, \quad j = 0,..,n$$

Metodo QR-LS

Si suppone che H abbia rango massimo.

$$\min S = \|Ha - y\|_2^2$$

- Applica una fattorizzazione QR alla matrice del sistema H=QR

$$Q \quad ortogonale \quad R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

R_1 triangolare superiore non singolare

Metodo QR-LS

$$\|Ha - y\|_2^2 = \|QRa - y\|_2^2 = \|Ra - Q^T y\|_2^2 =$$

$$= \|Ra - c\|_2^2, \quad c := Q^T y$$

- Si partiziona il vettore c:

$$c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \quad Ra - c = \begin{bmatrix} R_1 a - c_1 \\ -c_2 \end{bmatrix}$$

Metodo QR-LS

$$\min\|Ha-y\|_2^2 = \min\|Ra-c\|_2^2 =$$

$$= \min(\|R_1 a - c_1\|_2^2 + \|c_2\|_2^2) =$$

$$= \|c_2\|_2^2 + \min\|R_1 a - c_1\|_2^2$$

- Si risolve il sistema triangolare:

$$R_1 a = c_1$$

- Il residuo è dato da

$$\min\|Ha-y\|_2^2 = \|c_2\|_2^2$$

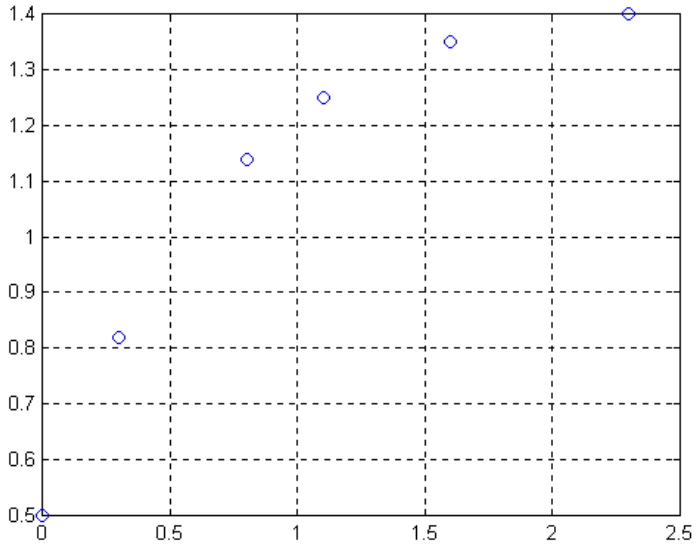
ESEMPIO 1

Siano date le misure di una certa quantità per diversi valori di tempo t:

» t=[0 0.3 0.8 1.1 1.6 2.3]';

» y=[0.5 0.82 1.14 1.25 1.35 1.40]';

» plot(t,y,'o'), grid on



Si vogliono approssimare i dati mediante un polinomio di grado 2 di incognite a_0, a_1, a_2 che minimizzi la somma dei quadrati degli scostamenti dei dati dal modello di approssimazione.

Per risolvere il problema LS si imposta il sistema lineare sovradeterminato Ha=y

ESEMPIO 1

rappresentato dalla matrice coefficienti H:

```
» H=[ones(size(t)) t t.^2]

H =
    1.0000         0         0
    1.0000    0.3000    0.0900
    1.0000    0.8000    0.6400
    1.0000    1.1000    1.2100
    1.0000    1.6000    2.5600
    1.0000    2.3000    5.2900
```

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \\ 1 & t_5 & t_5^2 \\ 1 & t_6 & t_6^2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

Il metodo utilizzato da MATLAB per risolvere il sistema sovradeterminato Ha=y è QR-LS cioè via fattorizzazione QR di H (richiamato dall'operatore \).

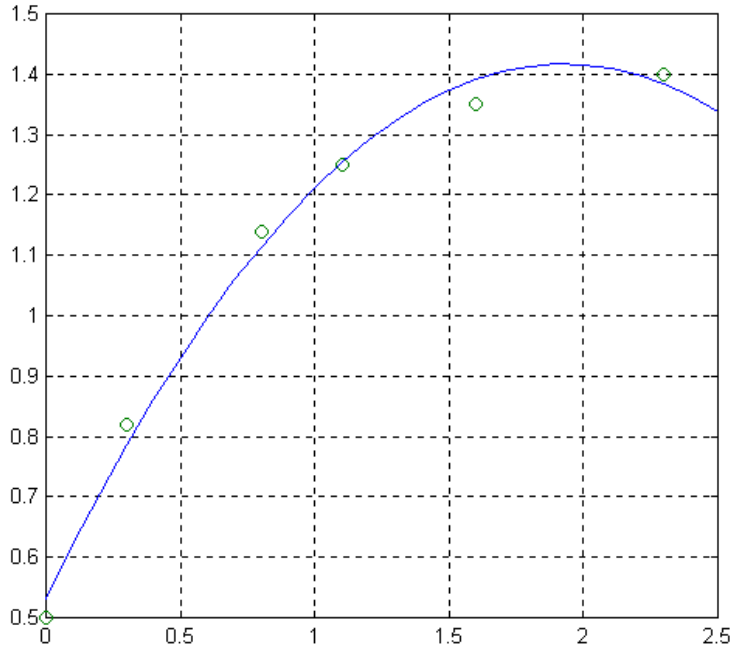
```
» a=H\y

a =    [0.5318    0.9191   -0.2387]'
```

ESEMPIO 1

Il polinomio di grado 2 che approssima i dati è il seguente:

$$y = 0.5318 + 0.9191t - 0.2387t^2$$

Visualizziamo quindi il polinomi discretizzazione dell'intervallo iniziali.




```
» T=(0:0.1:2.5)';
» Y=[ones(size(T)) T T.^2]*a;
» plot(T,Y,'-',t,y,'o'); grid on
```

ESEMPIO 2

Si vogliono ora approssimare i medesimi dati mediante la funzione esponenziale

$$y = a_0 + a_1 e^{-t} + a_2 t e^{-t}$$

di incognite a_0, a_1, a_2 che minimizzi la somma dei quadrati degli scostamenti dei dati dal modello di approssimazione. Si ottiene il seguente sistema sovradeterminato: Ha=y, rappresentato dalla matrice coefficienti H:

```
» H=[ones(size(t)) exp(-t)  t.*exp(-t)];
```

La soluzione si determina mediante l'operatore MATLAB backslash: \ (che richiama in questo caso QR-LS)

```
» a=H\y
```

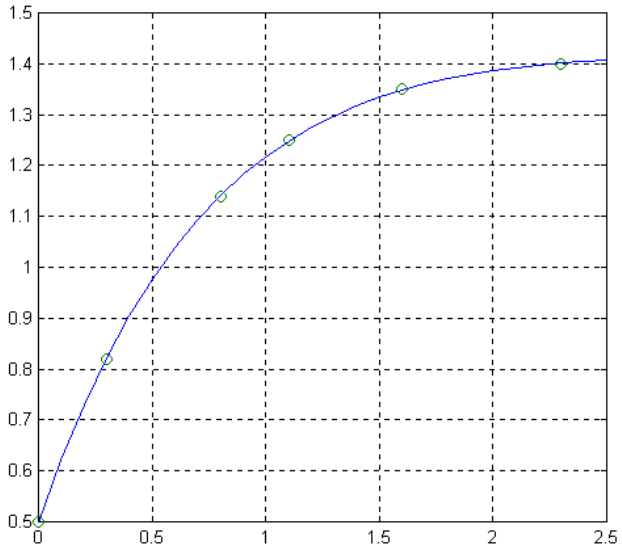
a =[1.3974 -0.8988 0.4097]

La funzione approssimante i dati iniziali è il seguente:

$$y = 1.3974 - 0.8988e^{-t} + 0.4097te^{-t}$$

Visualizziamo quindi la funzione approssimante per una discretizzazione dell'intervallo di definizione dei dati iniziali.

```
» T=(0:0.1:2.5)';
» Y=[ones(size(T)) exp(-T) T.*exp(-T)]*a
» plot(T,Y,'-',t,y,'o');grid on
```



Metodo della decomposizione in valori singolari per il problema LS (SVD-LS)

TEOREMA

Sia $H \in C^{mxn}$ di rango k, con $m \geq n \geq k$

e sia $H = U\Sigma V^T$ la decomposizione di H.

Allora la soluzione di minima norma del problema dei minimi quadrati e' data da

$$a^* = \sum_{i=1}^{k} \frac{u_i^H y}{\sigma_i} v_i$$

ed il valore minimo e':

$$\gamma^2 = \sum_{i=k+1}^{n} \left| u_i^H y \right|^2$$

METODO SVD-LS

Dimostrazione

$$\|Ha-y\|_2^2 = \|U^H(Ha-y)\|_2^2$$ Poichè U è ortogonale

$$= \|U^H HVV^H a - U^H y\|_2^2$$

$$\|Ha-y\|_2^2 = \|\Sigma z - U^H y\|_2^2$$ Posto $z=V^H a$:

$$= \sum_{i=1}^{n} \left|\sigma_i z_i - u_i^H y_i\right|^2$$

$$= \sum_{i=1}^{k} \left|\sigma_i z_i - u_i^H y_i\right|^2 + \sum_{i=k+1}^{n} \left|u_i^H y_i\right|^2$$

METODO SVD-LS

Il minimo si ha per

$$z_i = \begin{cases} \dfrac{u_i^H y}{\sigma_i}, & i = 1,..,k \\ qualunque & i = k+1,..,n \end{cases}$$

Se k<n la soluzione non e' unica, quella di minima norma e' data da

$$z_i = \begin{cases} \dfrac{u_i^H y}{\sigma_i}, & i = 1,..,k \\ 0 & i = k+1,..,n \end{cases}$$

Metodo SVD-LS

Poiche' a=Vz, e

$$\|a^*\| = \|y^*\|$$

$$a^* = Vz^* = \sum_{i=1}^{k} z_i^* v_i = \sum_{i=1}^{k} \frac{u_i^H y}{\sigma_i} v_i$$

$$\gamma^2 = \|Ha - y\|_2^2 = \sum_{i=k+1}^{n} \left| u_i^H y \right|^2$$

#

Metodo SVD-LS

```
function a=svdLS(H,y,tol)
[m,n]=size(H);
if m<n     disp('errore: num col > num righe');return
end
z=zeros(n,1);
a=zeros(n,1);
% Decomposizione SVD; estrazione dei valori singolari
[U,S,V]=svd(H);
sigma=diag(S,0)
d=U'*y;
i=sigma>tol;
z(i)=d(i)./sigma(i); % solo per valori singolari >tol
disp('condizionamento del problema')
cond=sigma(1)/sigma(n)
a=V*z;
```

Metodo SVD-LS

Condizionamento del problema $\frac{\sigma_1}{\sigma_n}$

Valori singolari piccoli sono indice di malcondizionamento del problema.

Tipicamente si usa una tolleranza sotto la quale si considera il valore singolare numericamente nullo.

Sia TOLL tale tolleranza, allora la soluzione al problema LS diviene:

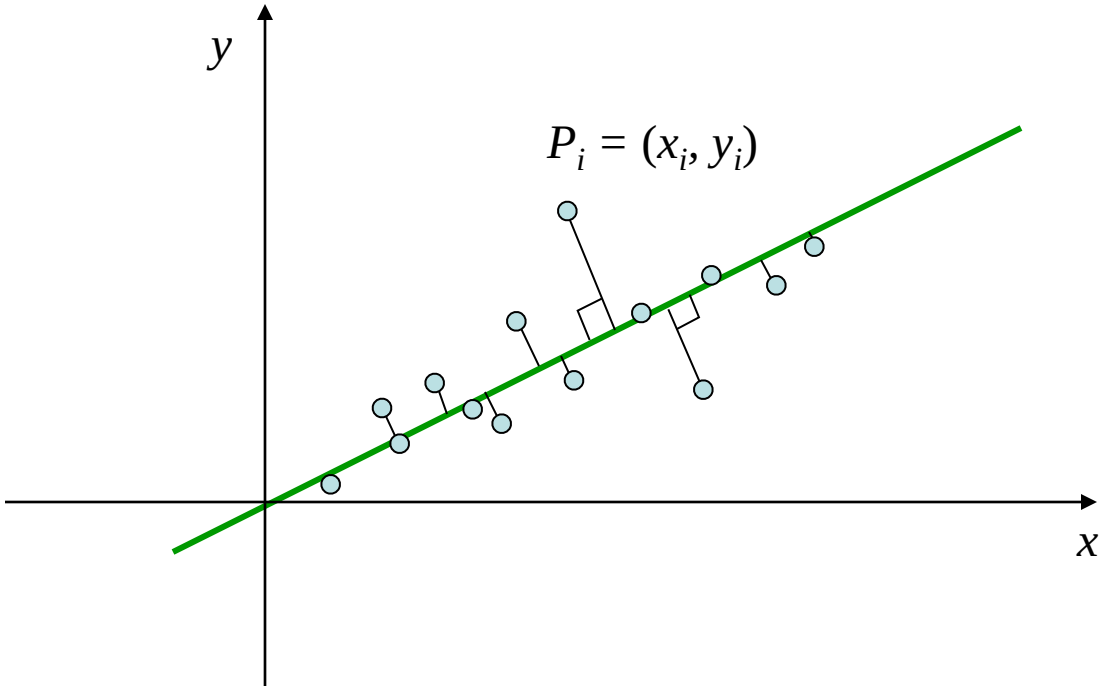
$$z_i = \begin{cases} \dfrac{u_i^H y}{\sigma_i}, & se \quad \sigma_i > TOLL \\ 0 & se \quad \sigma_i \leq TOLL \end{cases}$$

Principal Component Analysis PCA

usage of spectral analysis to analyze the shape and dimensionality of scattered data

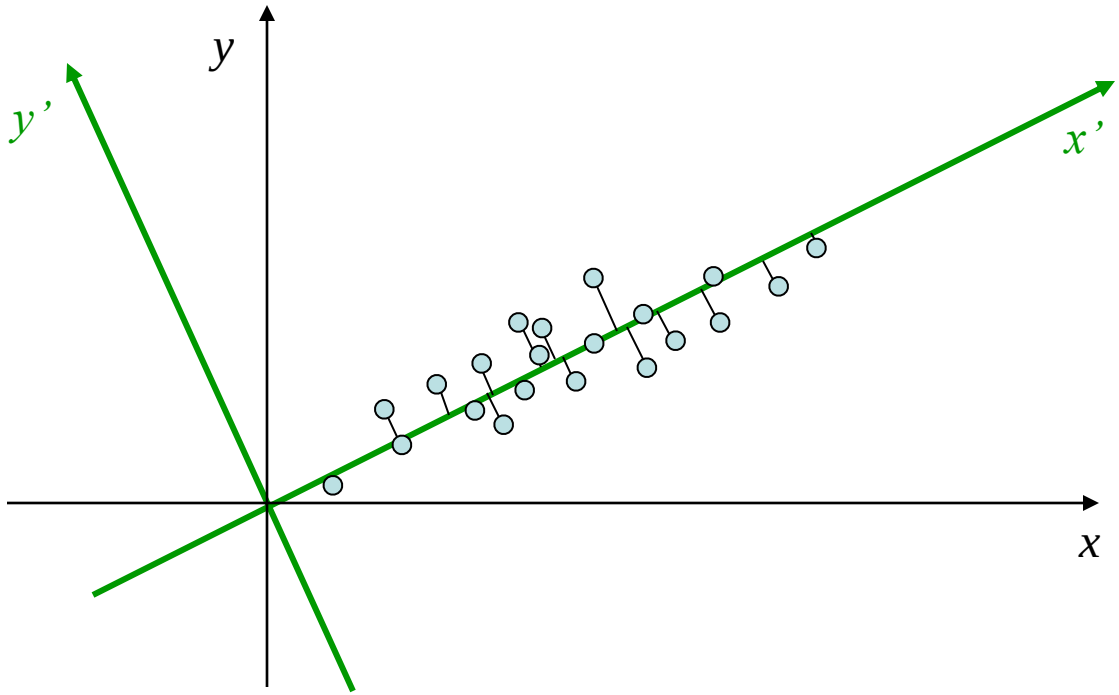
Line fitting

- Orthogonal offsets minimization



PCA – the general idea

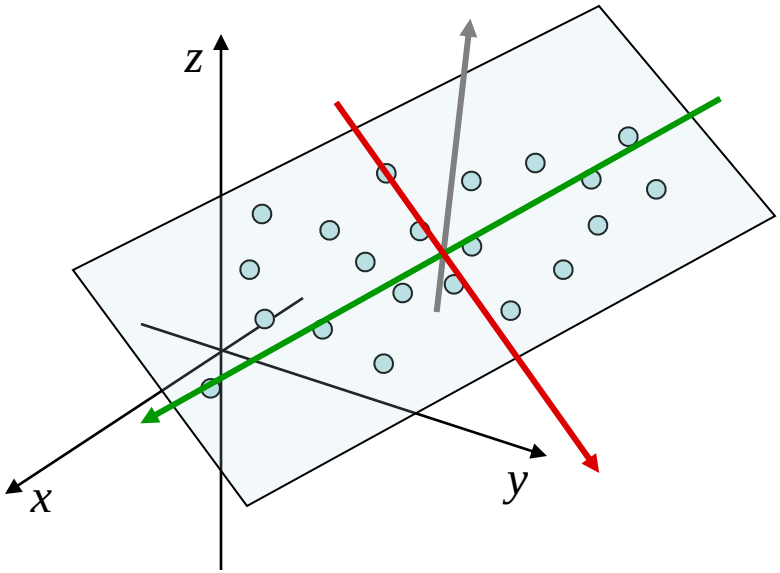
- PCA finds an orthogonal basis that best represents given data set.



- The sum of distances^2 from the x' axis is minimized.

PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.



3D point set in standard basis

- PCA finds a best approximating plane (again, in terms of $\Sigma distances^2$)

Notations

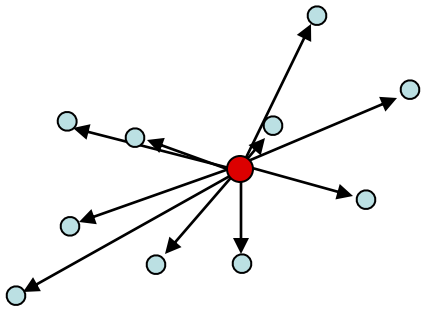
- Denote our data points by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in R^d$

$$\boldsymbol{x}_1 = \begin{pmatrix} x_1^1 \\ x_1^2 \\ \vdots \\ x_1^d \end{pmatrix}, \quad \boldsymbol{x}_2 = \begin{pmatrix} x_2^1 \\ x_2^2 \\ \vdots \\ x_2^d \end{pmatrix}, \quad \ldots, \quad \boldsymbol{x}_n = \begin{pmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^d \end{pmatrix}$$

The origin of the new axes

- The origin is zero-order approximation of our data set (a point)
- It will be the center of mass:

$$\boldsymbol{m} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$$



- It can be shown that:

$$\boldsymbol{m} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{x}\|^2$$

Covariance matrix

- Denote $\boldsymbol{y}_i = \boldsymbol{x}_i - \boldsymbol{m}, \quad i = 1, 2, \ldots, n$

$$\mathbf{S} = \sum_{k=1}^{n} \boldsymbol{y}_k \boldsymbol{y}_k^{\mathrm{T}}$$

$$\sum_{k=1}^{n} \begin{pmatrix} y_k^1 \\ y_k^2 \\ \vdots \\ y_k^d \end{pmatrix} \begin{pmatrix} y_k^1 & y_k^2 & \cdots & y_k^d \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} y_k^i y_k^j \end{pmatrix}$$

$d \times d$

Technical remarks:

- $\boldsymbol{S} = XX^T$, X is $d \times n$ matrix $\quad X = \begin{pmatrix} x^1 - m & x^2 - m & \dots & x^d - m \end{pmatrix}$

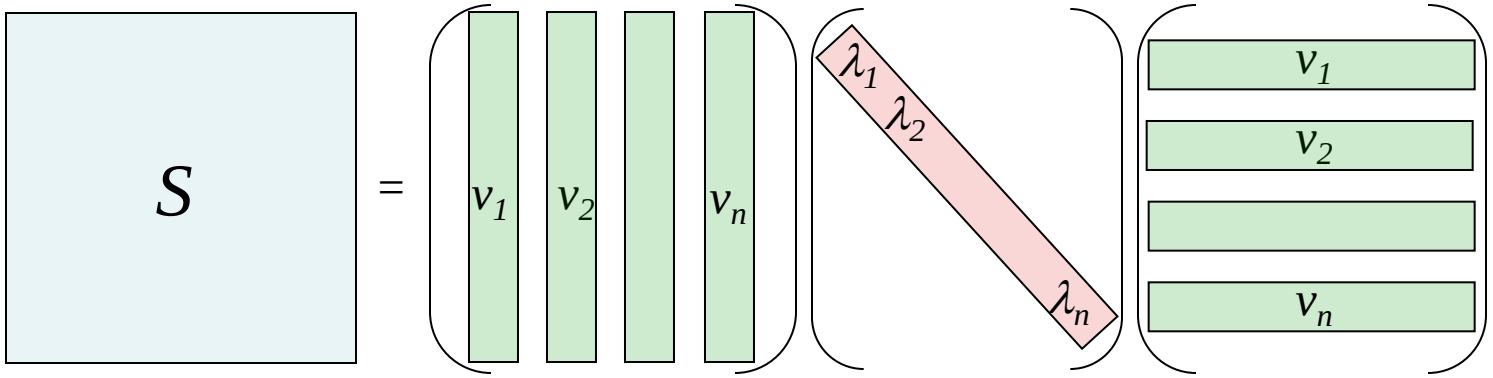
$$\sum_{k=1}^{n} \begin{pmatrix} y_k^1 \\ y_k^2 \\ \vdots \\ y_k^d \end{pmatrix} \begin{pmatrix} y_k^1 & y_k^2 & \dots & y_k^d \end{pmatrix} = \underbrace{\begin{pmatrix} & & \\ & \sum_{k=1}^{n} y_k^i y_k^j & \\ & & \end{pmatrix}}_{d \times d} =$$

$$= \begin{pmatrix} y_1^1 & y_2^1 & \cdots & y_n^1 \\ y_1^2 & y_2^2 & & y_n^2 \\ \vdots & \vdots & & \vdots \\ y_1^d & y_2^d & \cdots & y_n^d \end{pmatrix} \begin{pmatrix} y_1^1 & y_1^2 & & y_1^d \\ y_2^1 & y_2^2 & & y_2^d \\ \vdots & \vdots & & \vdots \\ y_n^1 & y_n^2 & & y_n^d \end{pmatrix} = XX^T$$

Covariance matrix - eigendecomposition

$\boldsymbol{S}$ is symmetric

$\Rightarrow$ $\boldsymbol{S}$ has eigendecomposition: $\boldsymbol{S} = V\Lambda V^{\mathrm{T}}$

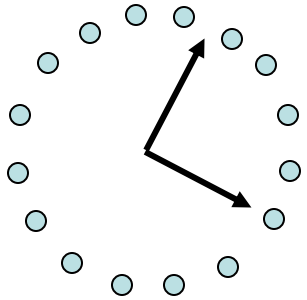


The eigenvectors form orthogonal basis

Principal components

- The eigenvectors of S identify a new orthogonal coordinate system.
- ***S*** measures the “scatterness” of the data.
- Eigenvectors that correspond to big eigenvalues are the directions in which the data has strong components.
- the axis v1 is selected as the direction of maximum variance
- If the eigenvalues are more or less the same – there is not preferable direction.

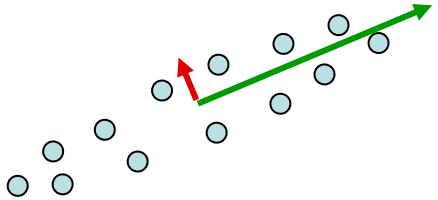
Principal components



- There's no preferable direction
- $\boldsymbol{S}$ looks like this:

$$\begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix}$$

- Any vector is an eigenvector



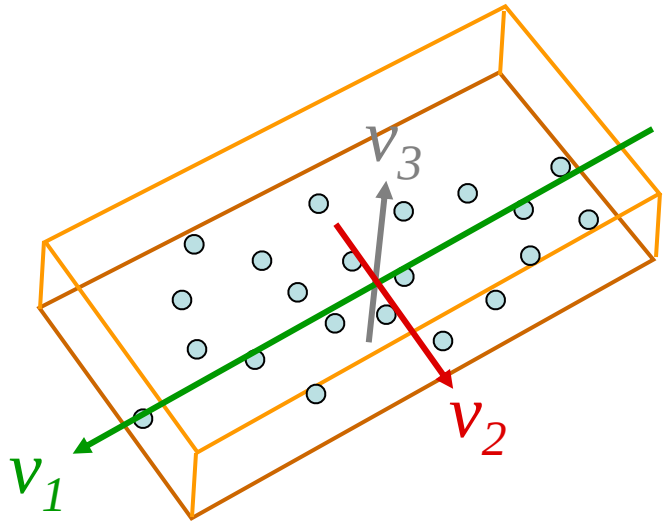
- There is a clear preferable direction
- $\boldsymbol{S}$ looks like this:

$$\boldsymbol{V}\begin{pmatrix} \lambda & \\ & \mu \end{pmatrix}\boldsymbol{V}^T$$

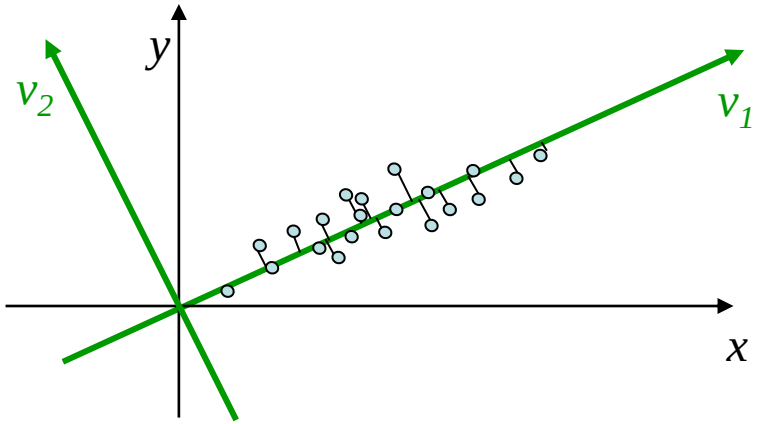
- μ is close to zero, much smaller than λ.

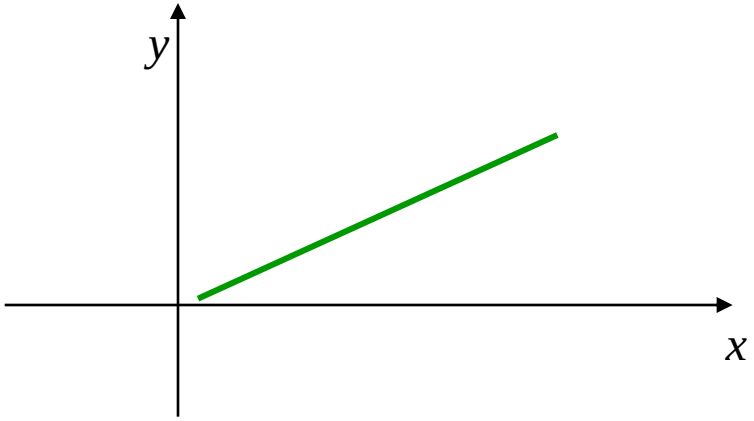
How to use what we got

- For finding oriented bounding box – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the mean point $\boldsymbol{m}$.

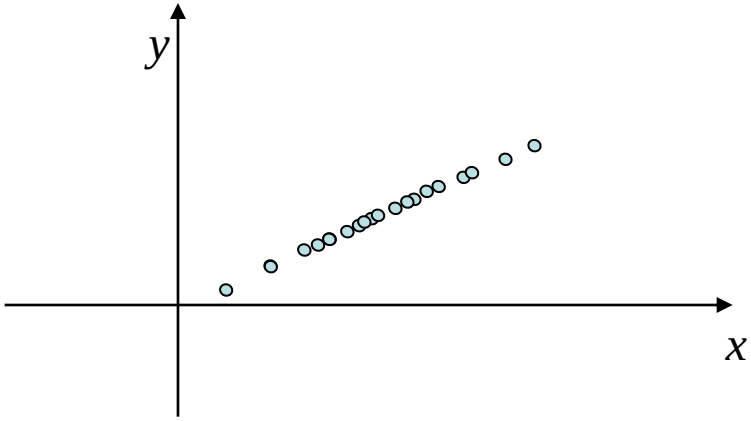


For approximation



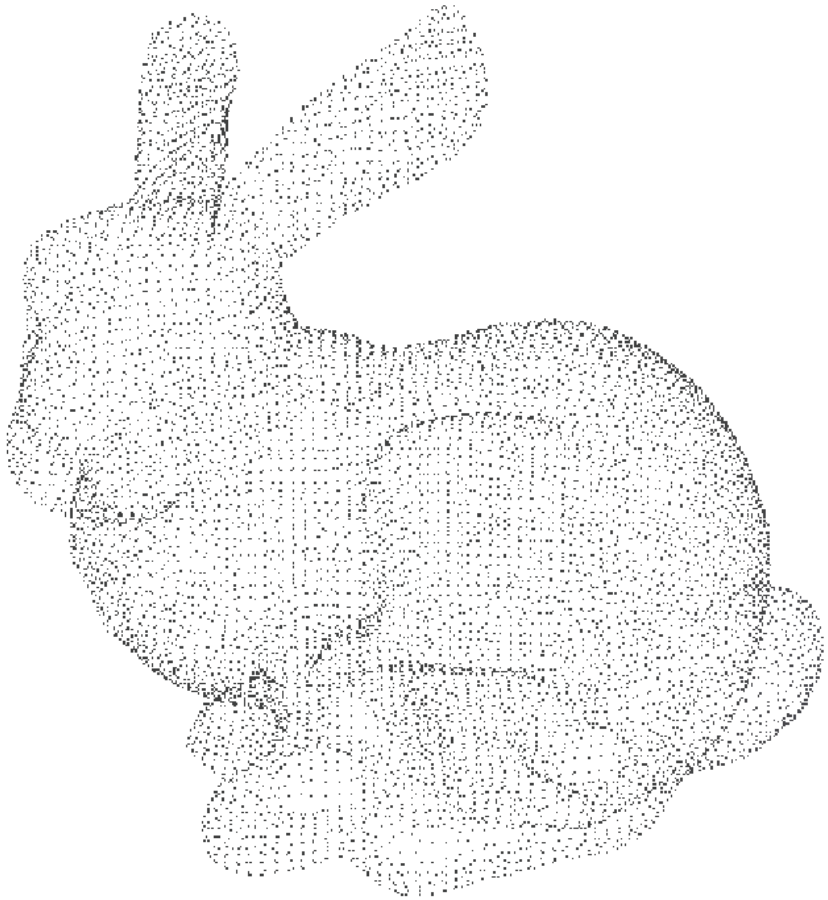


This line segment approximates the original data set



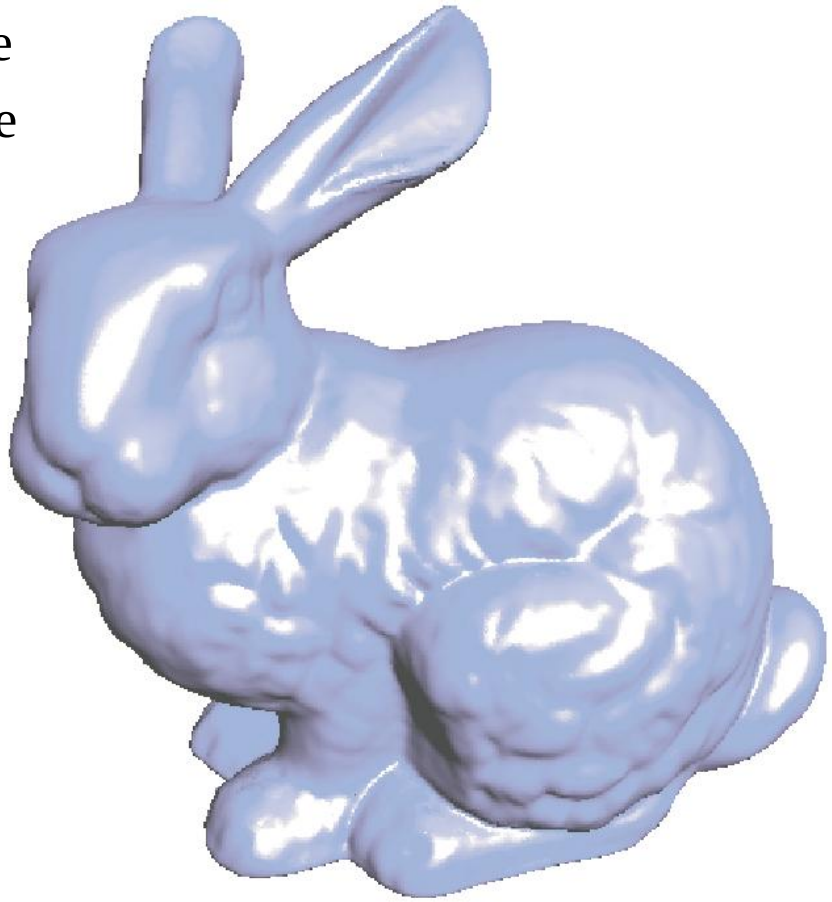
The projected data set approximates the original data set

Local surface fitting to 3D points



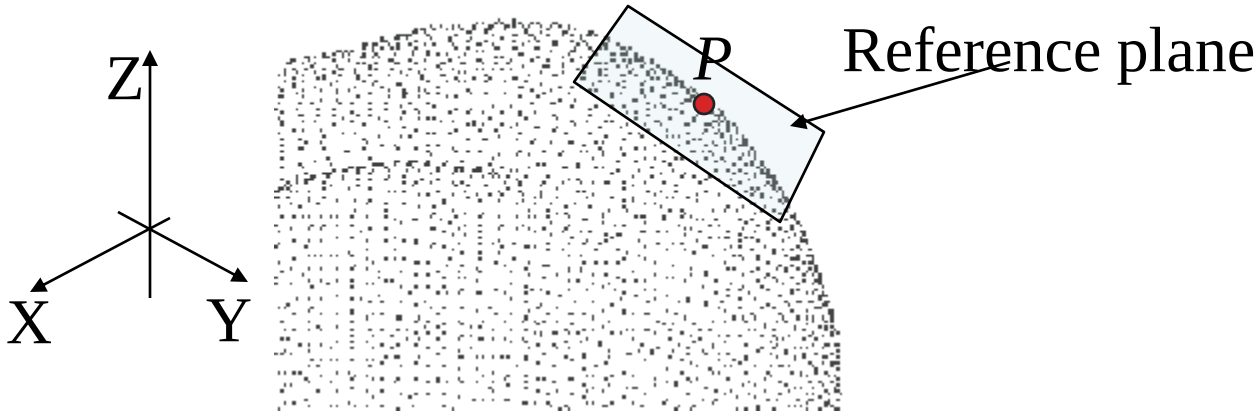
Local surface fitting to 3D points

Locally approximate
a polynomial surface
from points



Fitting local polynomial

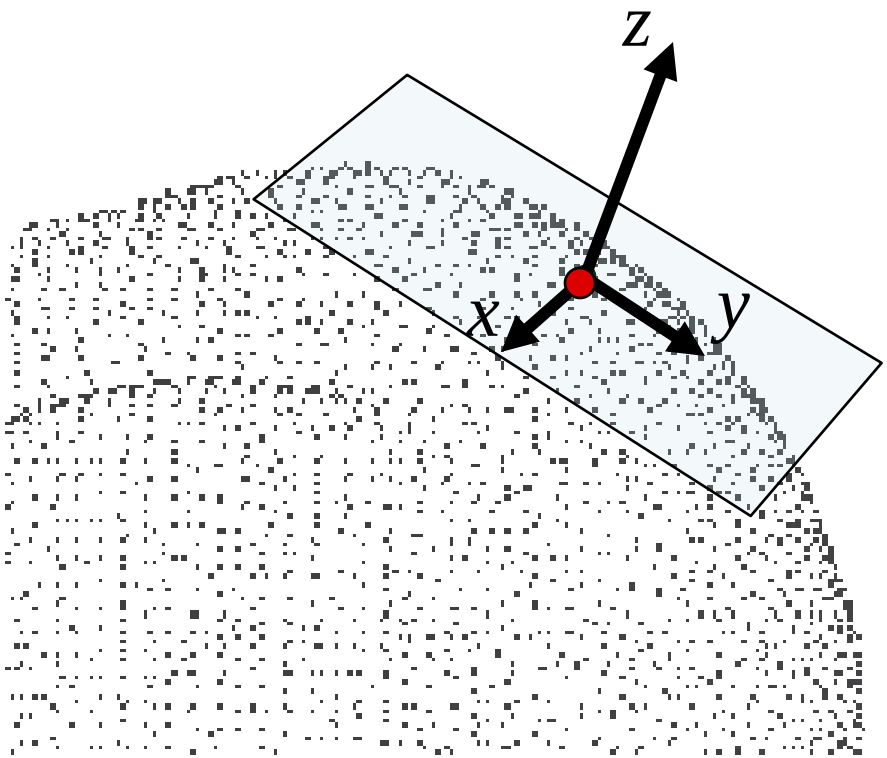
- Fit a local polynomial around a point *P*



Fitting local polynomial surface

STEP 1 Compute a reference plane that fits the points close to P

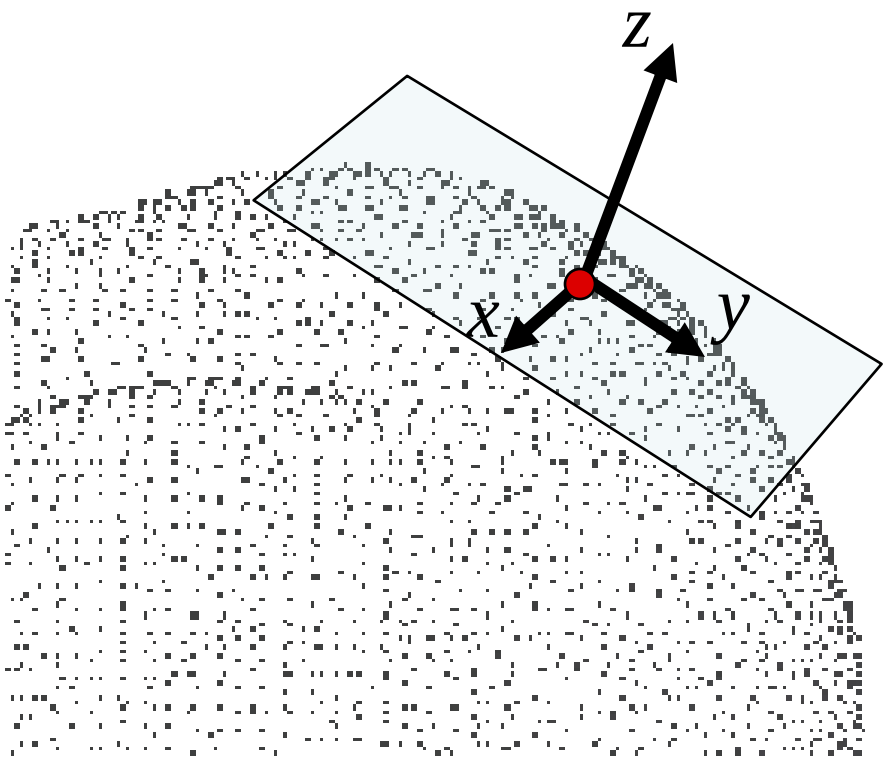
Use the local basis defined by the normal to the plane!



Fitting local polynomial surface

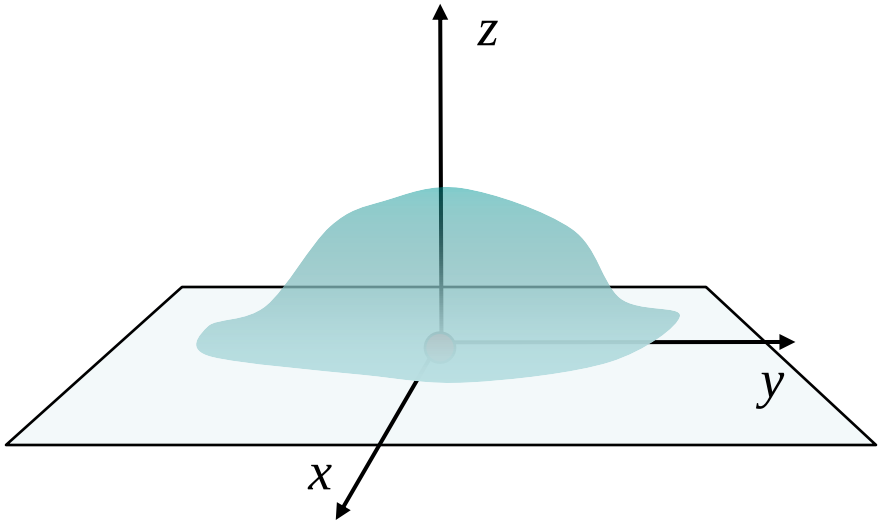
STEP 2

Fit polynomial $z = p(x,y) = ax^2 + bxy + cy^2 + dx + ey + f$



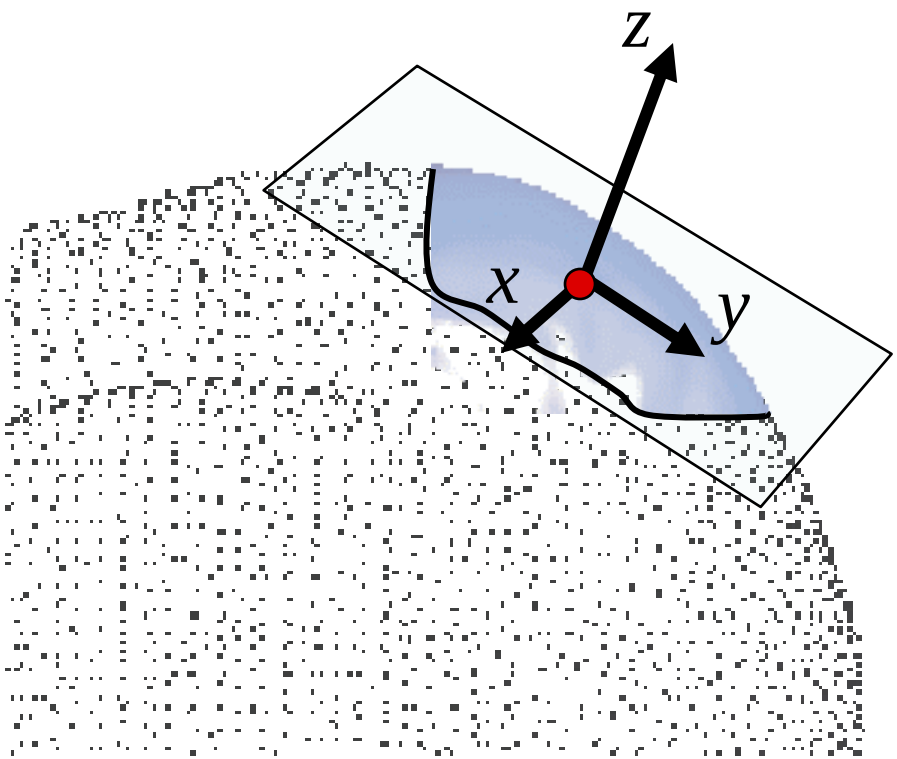
Fitting local polynomial surface

- Fit polynomial $z = p(x,y) = ax^2 + bxy + cy^2 + dx + ey + f$



Fitting local polynomial surface

- Fit polynomial $z = p(x,y) = ax^2 + bxy + cy^2 + dx + ey + f$



Fitting local polynomial surface

- Again, solve the system in LS sense:

$$ax_1^2 + bx_1y_1 + cy_1^2 + dx_1 + ey_1 + f = z_1$$
$$ax_2^2 + bx_2y_2 + cy_2^2 + dx_2 + ey_2 + f = z_2$$
$$\ldots$$
$$ax_n^2 + bx_ny_n + cy_n^2 + dx_n + ey_n + f = z_n$$

- Minimize $\Sigma \, \|z_i - p(x_i, y_i)\|^2$

Fitting local polynomial surface

- Also possible (and better) to add weights:

$$\min_{p \in \Pi^2} \sum_{j=1}^{N} \left(p(x_i, y_i) - z_i \right)^2 w_i \qquad w_i > 0$$

- The weights get smaller as the distance from the origin point grows.

Moving Least Squares (MLS)

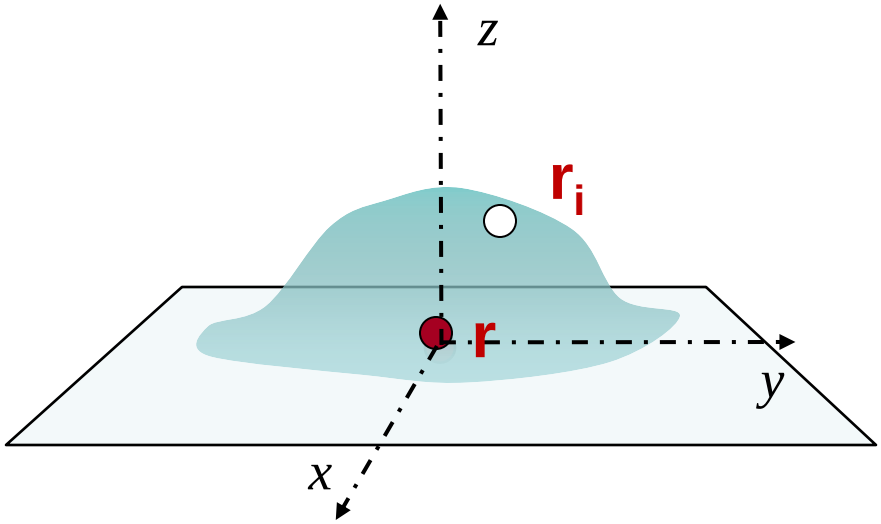
- Instead of **scalar weights** consider **weigth functions**

$$w_i(r) = \frac{e^{-a\|r-r_i\|^2}}{\|r-r_i\|^2}$$

- the weights are computed from the distances of the ri to r using a smooth, positive, monotone decreasing function, the gaussian

$$\min_{p\in\Pi^2} \sum_{j=1}^{N} \left(p(x_i, y_i) - z_i \right)^2 e^{-\frac{\|r_i - r\|^2}{h^2}}$$

Non-negative Weight function



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Serena Morigi

Dipartimento di Matematica

morigi@dm.unibo.it

http://www.dm.unibo.it/~morigi