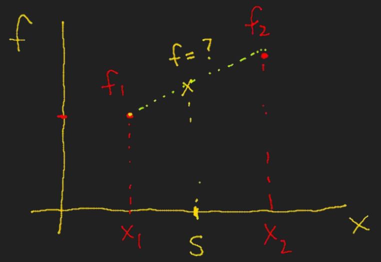
Lecture 3

2020-07-09

Interpolation

Interpolation



e.g. time   room temp

Given x_1, f_1 ← discrete data

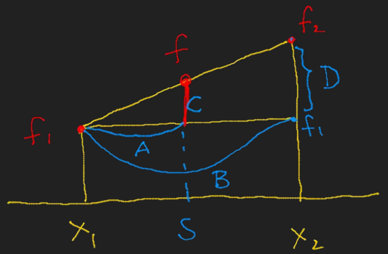
x_2, f_2 do not have a functional relation among the data

s

$\boxed{\text{what is } f \text{ at } x = s \text{ ?}}$

Assume Linear variation of f between x_1, x_2

① geometric view



$$\frac{C}{D} = \frac{A}{B}$$

$$C = f - f_1$$
$$D = f_2 - f_1$$
$$A = s - x_1$$
$$B = x_2 - x_1$$

$$\frac{f - f_1}{f_2 - f_1} = \frac{s - x_1}{x_2 - x_1}$$

$$\Rightarrow \quad f = \underbrace{f_1 + \frac{f_2 - f_1}{x_2 - x_1}(s - x_1)}$$

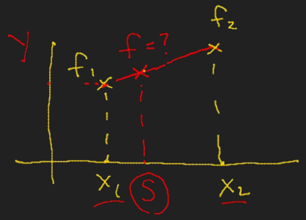
② algebraic view

linear eqn. between x_1, x_2

$$y = ax + b$$

$a = ?$

$b = ?$



$$\begin{cases} \text{at } x_1: & f_1 = a x_1 + b \\ \text{at } x_2: & f_2 = a x_2 + b \end{cases}$$

Solve for a, b

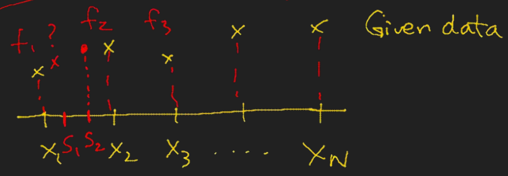
$$a = \frac{f_2 - f_1}{x_2 - x_1}, \quad b = \frac{f_1 x_2 - f_2 x_1}{x_2 - x_1}$$

$$y = \left(\frac{f_2 - f_1}{x_2 - x_1}\right) \cdot x + \left(\frac{f_1 x_2 - f_2 x_1}{x_2 - x_1}\right)$$

when

$x = S$

$$f = y = \left(\frac{f_2 - f_1}{x_2 - x_1}\right) \cdot S + \left(\frac{f_1 x_2 - f_2 x_1}{x_2 - x_1}\right)$$



$x = [x_1, x_2, x_3, \cdots, x_N]$

$f = [f_1, f_2, f_3, \cdots, f_N]$

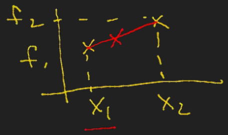
$S = [S_1, S_2, \cdots, S_M]$

Given

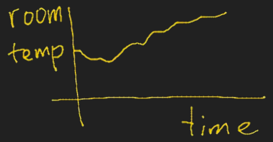
$\Rightarrow f_x = [\ ?,\ ?,\ \cdots\ ?\ ]$

1-D interpolation

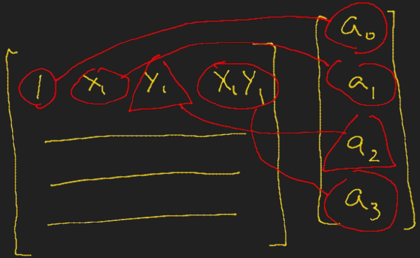
dimensional



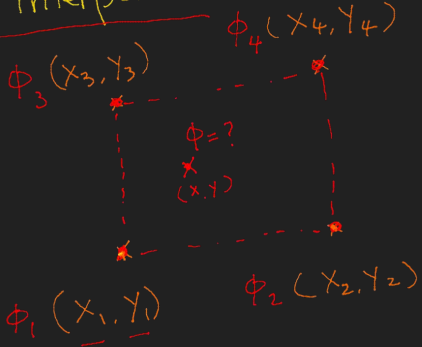
e.g.



$$Y = ax+b$$



2-D interpolation



bi-Linear relation

$$\phi = a_0 + a_1 x + a_2 y + a_3 xy$$

find $\boxed{a_0, a_1, a_2, a_3}$ from data

$$\begin{bmatrix} x_1, & y_1, & \phi_1 \\ x_2, & y_2, & \phi_2 \\ x_3, & y_3, & \phi_3 \\ x_4, & y_4, & \phi_4 \end{bmatrix}$$

$$\begin{array}{ll} (x_1, y_1): & \\ (x_2, y_2): & \\ (x_3, y_3): & \\ (x_4, y_4): & \end{array} \left\{ \begin{array}{l} a_0 + a_1 x_1 + a_2 y_1 + a_3 x_1 y_1 = \phi_1 \\ \underline{a_0} + \underline{a_1 x_2} + \underline{a_2 y_2} + \underline{a_3 x_2 y_2} = \phi_2 \\ a_0 + a_1 x_3 + a_2 y_3 + a_3 x_3 y_3 = \phi_3 \\ a_0 + a_1 x_4 + a_2 x_4 + a_3 x_4 y_4 = \phi_4 \end{array} \right] \Rightarrow$$

in "matrix" form:

unknowns

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix}$$

1 D interpolation in matrix form $\phi = a_0 + a_1 x$

$$\underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}}_{A\ \ 2\times2} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}}_{b}$$

Solve for $[a_0, a_1]$

2D interpolation in matrix form

$$\underbrace{\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix}}_{A\ \ 4\times4} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix}}_{b}$$

Solve for $[a_0, a_1, a_2, a_3]$

unknowns
$[a] \leftarrow$ numpy.linalg.solve(A, b)
np

3D interpolation

$\phi \to (x, y, z)$

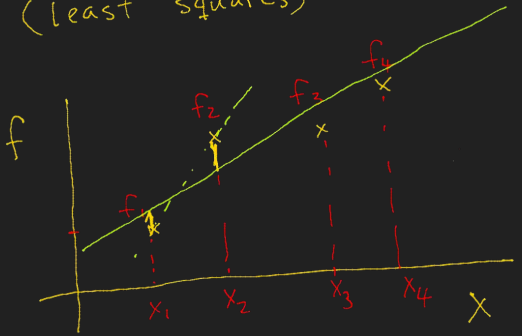
$$\phi = a_0 + a_1 x + a_2 y + a_3 z + a_4 xy + a_5 xz + a_6 yz + a_7 xyz$$

8 terms

$$\underbrace{\begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1 y_1 & x_1 z_1 & y_1 z_1 & x_1 y_1 z_1 \\ 1 & x_2 & y_2 & z_2 & x_2 y_2 & x_2 z_2 & y_2 z_2 & x_2 y_2 z_2 \\ \vdots & & & & & & & \\ 1 & x_8 & y_8 & z_8 & \cdots & \cdots & \cdots & \cdots \end{bmatrix}}_{A \quad 8\times 8} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_7 \end{bmatrix}}_{8\times 1} = \underbrace{\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_8 \end{bmatrix}}_{b \quad 8\times 1}$$

Linear regression (least squares)

Linear regression
(least squares)



"best" straight line

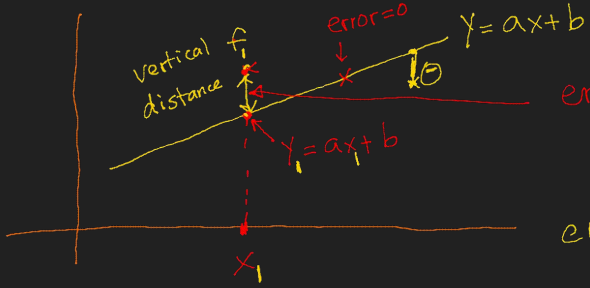
minimal error = ?

What is error?

There are many forms of errors.

Minimize total error

$$E = \sum_{i=1}^{N} e_i^2$$



$$\underset{e_1}{\text{error}_1} = f_1 - y_1 \quad (\sim\text{vertical distance})$$

$$= f_1 - (ax_1+b)$$

$$\underset{e_i}{\text{error}_i} = f_i - y_i$$

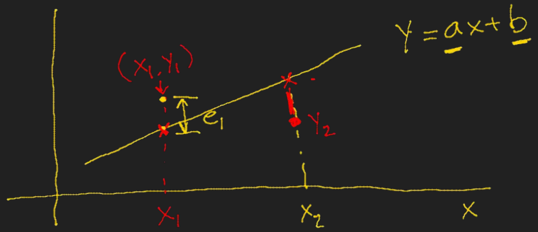
$i = 1, 2, \ldots$

Find a straight line such that error is the smallest.

total error

$$E = e_1^2 + e_2^2 + e_3^2 + \cdots + e_N^2$$

$$= \sum_{i=1}^{N} e_i^2$$



error

e_1 (at $x = x_1$) $= y_1 - Y$

$= y_1 - (ax_1 + b)$

at $x = x_2$

$e_2 = y_2 - Y$

$= y_2 - (ax_2 + b)$

~~total~~ total error

$E = e_1^2 + e_2^2 + \dots + e_N^2$

$= [y_1 - (ax_1 + b)]^2 + [y_2 - (ax_2 + b)]^2 + \dots + [y_N - (ax_N + b)]^2$

$\boxed{\text{Find } a, b \text{ that minimize } E}$

Consider only $e_1^2 = [y_1 - (ax_1 + b)]^2$

$= a^2 x_1^2 - 2ax_1(y_1 - b) + (y_1 - b)^2$

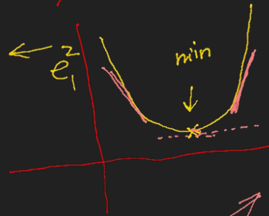
quadratic equation in x

2-nd order

$Y = ax + b$

$Y = ax^2 + b \leftarrow e_1^2$

$a > 0$



parabolic

$\boxed{\frac{de^2}{dx} = 0}$

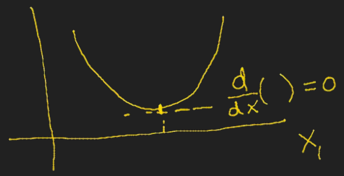
slope $= 0$

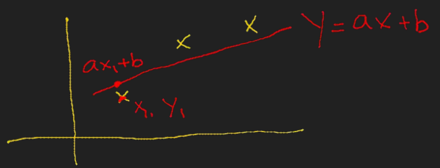
Find a, b

that satisfy $\frac{de_1^2}{dx} = 0$

$$E = [\overbrace{Y_1 - (ax_1+b)}^{e_1^2}]^2$$

$$= ax_1^2 - 2ax_1(Y_1-b) + (Y_1-b)^2$$





$$\frac{dE}{dx_1}=0$$

$$\frac{dE}{dx_1} = 2ax_1 - 2a(Y_1-b) = 0 \Rightarrow \boxed{ax_1+b=Y_1}$$ for e_1

Total $E = e_1^2 + e_2^2 + \dots + e_N^2$

$$\frac{dE}{dx_2}=0 \Rightarrow ax_2+b=Y_2$$

$$\frac{dE}{dx_3}=0 \Rightarrow ax_3+b=Y_3$$

$\vdots$

$$\frac{dE}{dY_N} \Rightarrow ax_N+b=Y_N$$

Find a,b by solving these equations

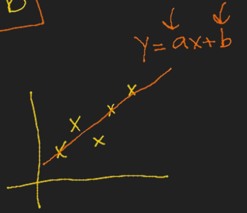
write in matrix form

$$N\underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}_{N\times 2}}_{A} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{2\times 1} = \underbrace{\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_N \end{bmatrix}_{N\times 1}}_{b}$$

numpy.linalg.lstsq(A,b)

Solve for

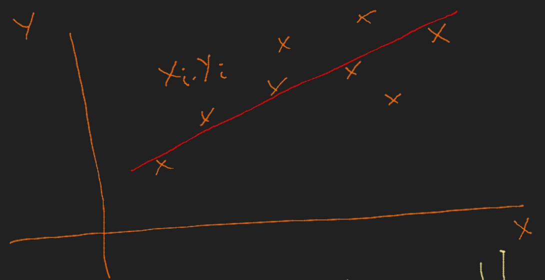
$\boxed{a, b}$



Numpy implementation

→ Given $\boxed{x_1, x_2, x_3, \ldots, x_N} \to X$ (numpy array)

$\boxed{y_1, y_2, y_3, \ldots, y_N} \to Y$



Form

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix}$$

$$S = \text{numpy.ones}(N) \to [\underbrace{1, 1, 1, \ldots, 1}_{N\ 1\text{'s}}]$$

$$\text{numpy.vstack}([X, S]) \to \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_N \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \begin{array}{l} X \\ S \end{array}$$

vertical

Transpose

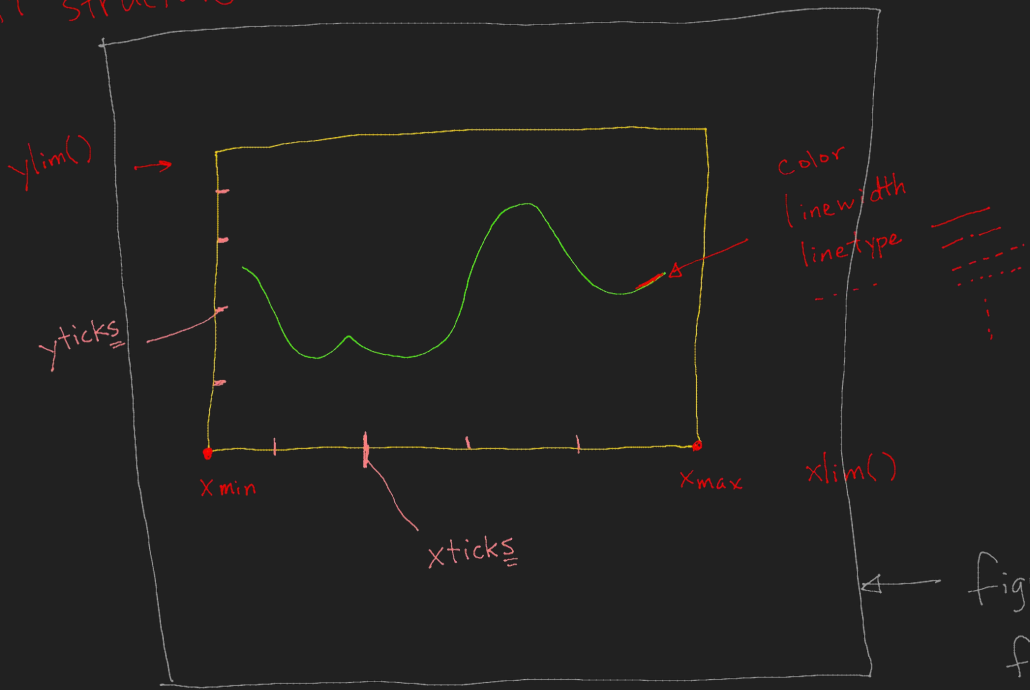
Capital "T"

$$A = \boxed{\text{np.vstack}([X, S]).T}$$

Call numpy.linalg.lstsq(A, b)

Matplotlib plotting

Polt structure

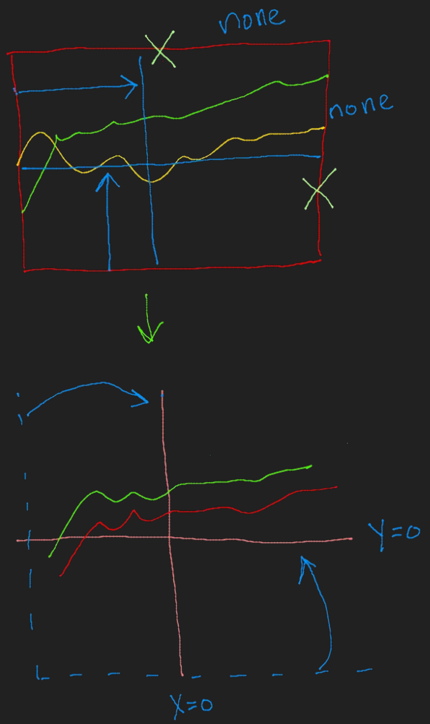


figsize

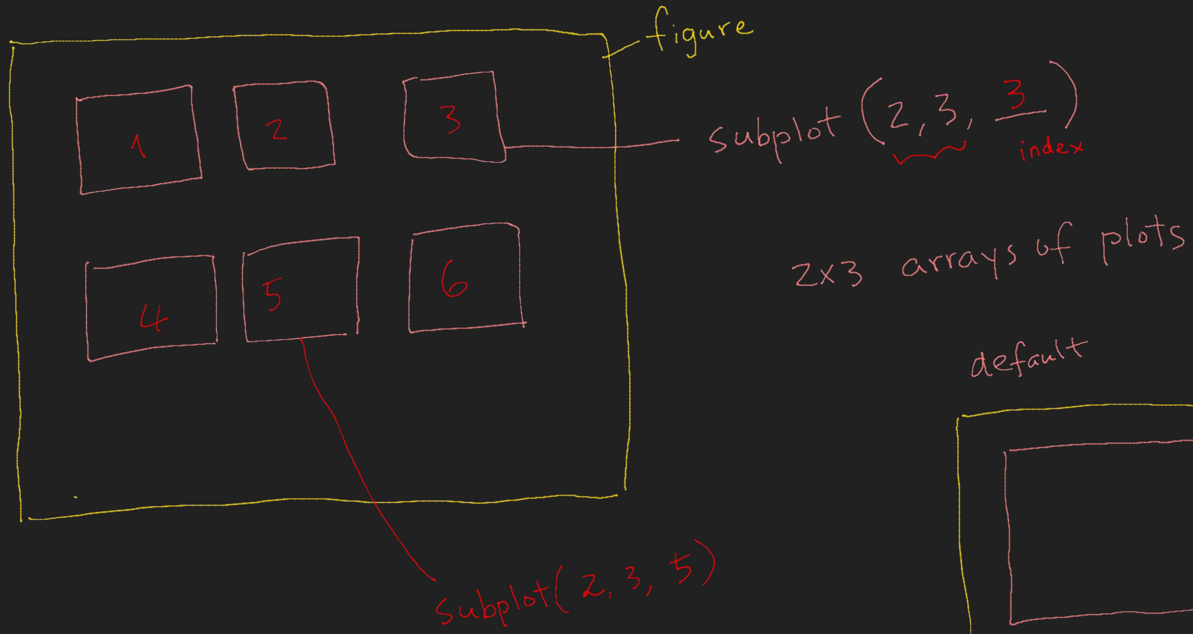
dpi: density per inch

Spines

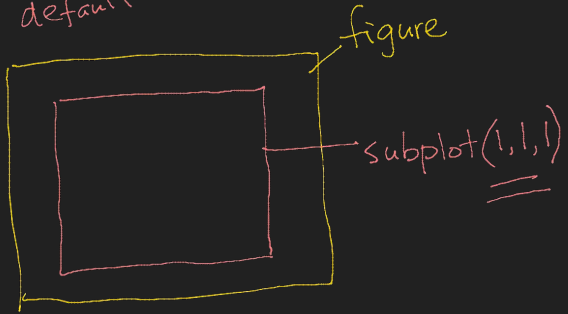
Normal plot
(default)



Subplot



default



see also Jupyter notebook plot.ipynb