Lagrange polynomials

$$L_k(x) = \frac{\prod_{j=0, j\neq k}^{n-1} (x - x_j)}{\prod_{j=0, j\neq k}^{n-1} (x_k - x_j)} \qquad k = 0, 1, \ldots n-1$$

denom depends only on interpolation points x_j's

num is polynomial in x, degree $n-1$

$L_k(x)$ has zeroes at x_j when $j \neq k$

$n=3$

$\Downarrow$

quadratic

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \qquad L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

$L_0(x_1) = L_0(x_2) = 0, \quad L_0(x_0) = 1$

$$L_k(x_j) = \delta_{kj} \qquad \delta_{kj} = \begin{cases} 1 & k=j \\ 0 & k\neq j \end{cases}$$

We use $L_k(x)$ as our basis functions

$$p(x) = \sum_{k=0}^{n-1} c_k \phi_k(x) = \sum_{k=0}^{n-1} c_k L_k(x)$$

but we require $p(x_j) = y_j$ by definition

$$p(x_j) = \sum_{k=0}^{n-1} c_k L_k(x_j) = \sum c_k \delta_{kj} = c_j = y_j$$

$$\Rightarrow p(x) = \sum_{k=0}^{n-1} y_k L_k(x)$$

[See barycentric formula for less operationally intensive implementation 6.2.2.2]

2nd Approximation Application: Linear Least-Squares Fitting

We have an input table of data points (x_j, y_j) for $j = 0, 1, \ldots \underline{N}-1$ where we assume that there is some uncertainty in each measured $y_j \Rightarrow \sigma_j$

So our function $p(x)$ will not be able to go through all the points (y_j) exactly

we try to get a $p(x)$ that does a 'good' job (there could be many that do a good job)

Number of data points N, larger than number of c_k params n $\Rightarrow$ $N > n$ overdetermined system

$$\underbrace{\begin{pmatrix} \phi_0(x_0) & \cdots & \phi_{n-1}(x_0) \\ \vdots & & \\ \phi_0(x_{N-1}) & \cdots & \phi_{n-1}(x_{N-1}) \end{pmatrix}}_{N \times n} \underbrace{\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix}}_{n \times 1} \approx \underbrace{\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}}_{N \times 1}$$

We can't solve $\Phi \vec{c} = \vec{y}$ exactly

We will try to get close. Minimize $\|\vec{y} - \Phi \vec{c}\|$

We still assume $p(x) = \sum_{k}^{n-1} c_k \phi_k(x)$

$p(x)$ is linear combination of c_k (not nec. of x)

To take into account measurement uncertainty

we minimize chi-squared statistic

$$\chi^2 = \sum_{j=0}^{N-1} \left[\frac{y_j - p(x_j)}{\sigma_j} \right]^2$$

weighted fit

and note $\chi^2 = \chi^2(c_k\text{'s})$

Find c_k's that minimize distance between

data (y_j) and theory $[p(x_j)]$ or model

weighted by error bar (σ_j)

minimize χ^2 wrt c_k

$$\frac{\partial \chi^2}{\partial c_k} = -2 \sum_j^{N-1} \left(\frac{y_j - p(x_j)}{\sigma_j^2} \right) \frac{\partial p(x_j)}{\partial c_k} = 0 \qquad k = 0, 1, \ldots n-1$$

We assume that each y_j is Gaussian distributed about a 'true' value with standard deviation σ_j

Straight-line fit

$$p(x) = c_0 + c_1 x \qquad n = 2$$

$$\chi^2 = \sum_j^{N-1} \left[\frac{y_j - c_0 - c_1 x_j}{\sigma_j} \right]^2$$

$$\frac{\partial \chi^2}{\partial c_0} = -2 \sum_j^{N-1} \left[\frac{y_j - c_0 - c_1 x_j}{\sigma_j^2} \right] = 0$$

$$\frac{\partial \chi^2}{\partial c_1} = -2 \sum_j^{N-1} \left[\frac{y_j - c_0 - c_1 x_j}{\sigma_j^2} \right] x_j = 0$$

Define $\quad S \equiv \sum_j^{N-1} 1/\sigma_j^2 \qquad S_x \equiv \sum^{N-1} x_j/\sigma_j^2 \qquad S_y \equiv \sum^{N-1} y_j/\sigma_j^2$

$$S_{xx} \equiv \sum^{N-1} x_j^2/\sigma_j^2 \qquad S_{xy} \equiv \sum^{N-1} x_j y_j/\sigma_j^2$$

$$\Delta \equiv S S_{xx} - S_x^2$$

We can compute all these from data

$$\sum \frac{y_j - c_0 - c_1 x_j}{\sigma_j^2} = 0 \quad \Rightarrow \quad S_y - c_0 S - c_1 S_x = 0$$

$$S_y = S c_0 + S_x c_1$$

$$\sum \left(\frac{y_j - c_0 - c_1 x_j}{\sigma_j^2} \right) x_j = 0 \quad \Rightarrow \quad S_{xy} = S_x c_0 + S_{xx} c_1$$

$$\begin{pmatrix} S & S_x \\ S_x & S_{xx} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} S_y \\ S_{xy} \end{pmatrix}$$

Solution $\quad c_0 = \dfrac{S_{xx} S_y - S_x S_{xy}}{\Delta} \qquad c_1 = \dfrac{S S_{xy} - S_x S_y}{\Delta}$

These are estimates of 'true' params so they have uncertainties

Recall our error propagation formula:

$$f = (x, y, \dots z) \qquad \sigma_f^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2 + \dots \left(\frac{\partial f}{\partial z}\right)^2 \sigma_z^2$$

$$\frac{\partial c_0}{\partial y_j} = \frac{1}{\Delta}\left[S_{xx} \frac{\partial S_y}{\partial y_j} - S_x \frac{\partial S_{xy}}{\partial y_j} \right] = \frac{1}{\Delta}\left[S_{xx}\left(1/\sigma_j^2\right) - S_x\left(x_j/\sigma_j^2\right) \right]$$

$$\frac{\partial c_1}{\partial y_j} = \frac{1}{\Delta}\left[S\left(x_j/\sigma_j^2\right) - S_x\left(1/\sigma_j^2\right) \right]$$

since c_0 and c_1 are functions of y_j

$$\sigma_{c_0}^2 = \sum \left(\frac{\partial c_0}{\partial y_j}\right)^2 \sigma_j^2$$

$$= \frac{1}{\Delta^2} \sum_j \left(\frac{S_{xx} - S_x x_j}{\sigma_j^2}\right)^2 \sigma_j^2$$

$$= \frac{1}{\Delta^2}\Big[S_{xx}^2 \underbrace{\sum 1/\sigma_j^2}_{S} - 2 S_{xx} S_x \underbrace{\sum x_j/\sigma_j^2}_{S_x} + S_x^2 \underbrace{\sum x_j^2/\sigma_j^2}_{S_{xx}} \Big]$$

$$= \frac{1}{\Delta^2}\left[S_{xx}^2 S - S_{xx} S_x^2 \right] = \frac{1}{\Delta^2} S_{xx} \big[\underbrace{S_{xx} S - S_x^2}_{\Delta} \big]$$

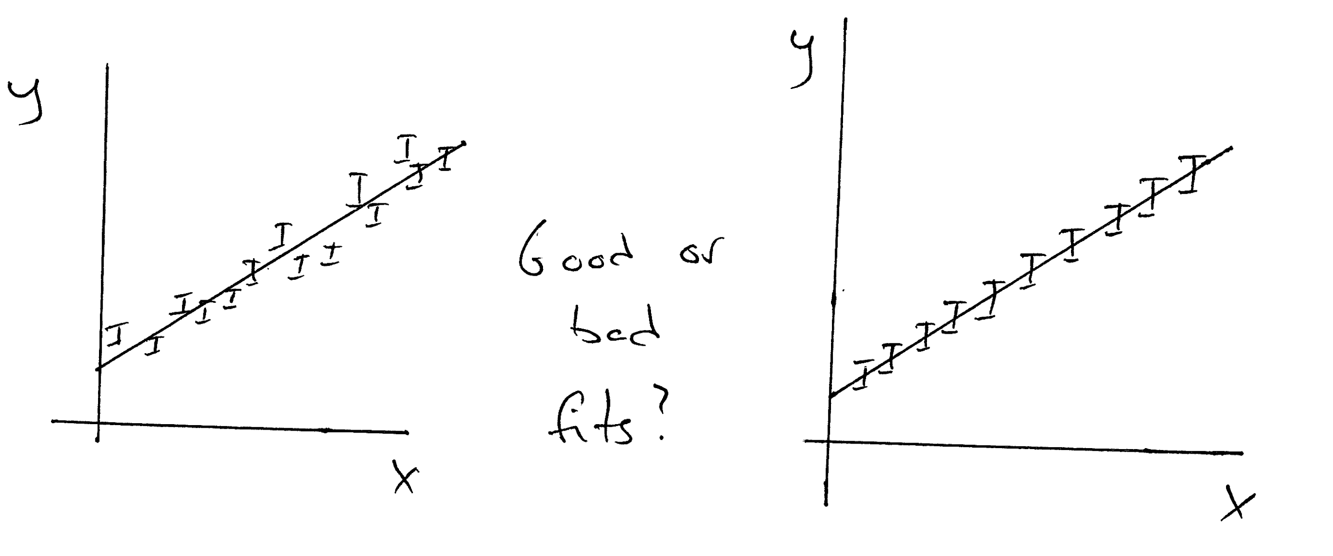
$$\sigma_{c_0}^2 = S_{xx}/\Delta \qquad \sigma_{c_1}^2 = S/\Delta \quad (\text{similar})$$

[example fit]

We chose c_0 and c_1 to minimize χ^2. We can now calculate its value

$$\chi^2 = \sum \left(\frac{y_j - c_0 - c_1 x_j}{\sigma_j} \right)^2$$

Gives information about 'goodness of fit'



If $y_j = c_0 + c_1 x_j$ for each j $\chi^2 = 0$

σ_j represents statistical spread of y_j about the 'true value'. $\chi^2 = 0$ is very very unlikely!

Expect each term in χ^2 to be about 1

Small correction based on # of fitting (free) params

$$\langle \chi^2 \rangle = \nu = N - m$$

ν = degrees of freedom

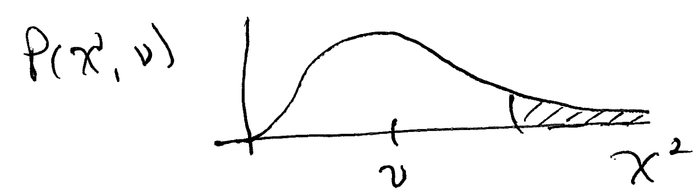
N = # data points
m = # fitting params

linear fit $\Rightarrow m = 2$

expect χ^2/ν (chi squared per degree of freedom) to be $\sim \underline{1}$

[Aside: You can find tables that give you probability of getting a certain value for $\chi^2, \nu \Rightarrow P(\chi^2, \nu)$

Tells you how often you expect to get a value larger than χ^2, ν if repeated many times



Ex $N = 10$, $m = 2$, $\nu = 8$

You find $\chi^2 = 2.7$ $P(2.7, 8) \cong 95\%$ errors too large?

$\chi^2 = 20$ $P(20, 8) = 1\%$ errors too small?

Using this information can help in deciding if you chose a 'good' model to fit the data (was linear better than quadratic?)

Sometimes the best model is non-linear. Typical physics examples

$$p(x) = C_0 e^{-C_1 x} \qquad p(x) = C_0 x^{C_1}$$

exponential power law

We will not cover non-linear least squares but you can linearize by hand

$$\underbrace{\ln p(x)}_{p'(x)} = \ln C_0 - C_1 x$$

$$p'(x) = C_0' + C_1' x$$

Suppose measured value for x_j is $y_j \pm \sigma_j$

$$\chi^2 = \sum \frac{1}{\delta_j^2} \left[\ln y_j - \ln C_0 + C_1 x_j \right]^2$$ where

$$\delta_j = \frac{\partial}{\partial y}(\ln y_j)\,\sigma_j = \sigma_j / y_j$$

Least squares fit returns $c_0' \pm \sigma_{c_0'}$ $\quad c_1' \pm \sigma_{c_1'}$

Q Find ~~[illegible]~~ $c_0 \pm \sigma_{c_0}$

$$c_0' = \ln c_0 \qquad c_0 = e^{c_0'} \qquad \sigma_{c_0} = e^{c_0'}\sigma_{c_0'} = c_0 \sigma_{c_0'}$$

Ex $\quad y = a e^{-bx}$ $\quad$ ($a = 20$ $\quad b = 0.3$ true values)

fit returns $\quad a' = 3.01 \pm 0.29$ $\quad$ for some sample data

$$a = e^{3.01} = 20.3 \qquad \sigma_a = 5.9$$

$$b' = -0.31 \pm 0.08 = b$$

Linear fit: Normal equations

We did 1st order fit. What about higher degree poly fits?

We start ~~[illegible]~~, with most general linear fit

$$p(x) = \sum_{k=0}^{n-1} c_k \phi_k(x)$$

for poly fit $\phi_k(x)$ are monomials we saw earlier $\{1, x, x^2, x^3, \dots x^k, \dots\}$

$$\chi^2 = \sum_j^{N-1} \frac{1}{\sigma_j^2}\left(y_j - \sum_i^{n-1} c_i \phi_i(x_j)\right)^2$$

$$\frac{\partial \chi^2}{\partial c_k} = -2 \sum_j^{N-1} 1/\sigma_j^2 \left(y_j - \sum_i^{n-1} c_i \phi_i(x_j)\right) \phi_k(x_j) = 0$$

$$\sum_j^{N-1} 1/\sigma_j^2 (y_j \phi_k(x_j)) = \underbrace{\sum_j^{N-1} 1/\sigma_j^2 \sum_i^{n-1}}\, c_i\, \phi_i(x_j) \phi_k(x_j)$$

$$= \sum_i^{n-1} \sum_j^{N-1} \frac{\phi_k(x_j)}{\sigma_j} \frac{\phi_i(x_j)}{\sigma_j} c_i$$

Define $A_{jk} = \dfrac{\phi_k(x_j)}{\sigma_j}$ $\quad b_j = \dfrac{y_j}{\sigma_j}$

Matrix A is $N \times n$ (typically N large, n small)

called design matrix

$\vec{b}$ is $N \times 1$ column vector of data/uncertainty

we can compute these just using input data

$$\sum_i^{n-1} \sum_j^{N-1} A_{jk} A_{ji} c_i = \sum_j^{N-1} A_{jk} b_j$$

$$\sum_i^{n-1} (A^T A)_{ki} c_i = (A^T b)_k$$ holds for each component k

$$A^T A \vec{c} = A^T \vec{b}$$

A is $N \times n$ $\quad A^T$ is $n \times N$

$A^T A$ is $n \times n$ $\quad \vec{c}$ is $n \times 1$ (coeffs vector)

We compute A and $\vec{b}$ from data

So we have simple linear system of equations

We can solve for $\vec{c}$ (numpy.linalg.solve)

Our original χ^2 with these defs is

$$\chi^2 = (b - Ac)^T (b - Ac)$$

Need to find uncertainties σ_{c_i}

$$\sigma_{c_i}^2 = \sum_j^{N-1} \left(\frac{\partial c_i}{\partial y_j}\right)^2 \sigma_j^2$$

$$A^T A c = A^T b \Rightarrow c = (A^T A)^{-1} A^T b$$

define $U = A^T A$ and $V = (A^T A)^{-1}$

$(A^T A^{-1})$ is $n \times n$ and $A^T b$ is $n \times 1$

$$c_i = \sum_K^{n-1} V_{iK} (A^T b)_K = \sum_K^{n-1} V_{ik} \sum_\ell^{N-1} A_{\ell k} b_\ell$$

$$= \underbrace{\sum_K^{n-1} V_{ik} \sum_\ell^{N-1} \frac{\phi_K(x_\ell)}{\sigma_\ell}}_{\text{indep of } y_j} \frac{y_\ell}{\sigma_\ell}$$

$$\frac{\partial c_i}{\partial y_j} = \sum_K^{n-1} V_{iK} \frac{\phi_K(x_j)}{\sigma_j^2}$$

$$\sigma_{c_i}^2 = \sum_K^{n-1} \sum_\ell^{n-1} V_{iK} V_{i\ell} \left(\sum_j^{N-1} \frac{\phi_K(x_j) \phi_\ell(x_j)}{\sigma_j^2} \right)$$

$$\left(\sum_j^{N-1} \frac{\phi_k(x_j)\phi_\ell(x_j)}{\sigma_j^2} \right) = \sum_j^{N-1} A_{jk} A_{j\ell} = (A^T A)_{k\ell}$$

$$= U_{k\ell}$$

$$\sigma_{c_i}^2 = \sum_k^{n-1} \sum_\ell^{n-1} V_{ik} V_{i\ell} U_{k\ell} \cancel{\neq \sum_\ell^{n-1} \sum_k^{n-1}}$$

$$= \sum_\ell^{n-1} V_{i\ell} \underbrace{\sum_k^{n-1} V_{ik} U_{k\ell}}_{V \text{ is } U^{-1} \text{ so product is } \mathbb{I}} = \sum_\ell^{n-1} V_{i\ell} \delta_{i\ell} = V_{ii}$$

$$\sigma_{c_i}^2 = V_{ii} = \left[(A^T A)^{-1} \right]_{ii}$$

diagonal elements of V are the squared uncertainty of coeffs

V is <u>covariance</u> matrix

depends on model ϕ_k and uncertainties σ_j

[See newnormal.py]