Lecture 1-1: Nonlinear Regression

with least-squares fitting

Zejian Li
(li.zejian@ictp.it)

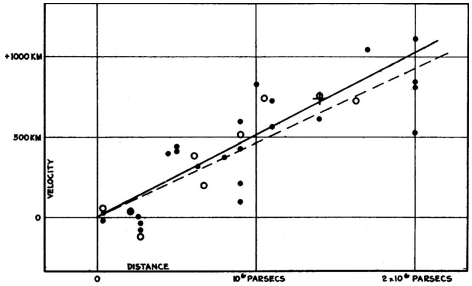
18 Oct. 2024

Last lecture: linear least-squares fitting

The process of constructing a parametrized linear function $f(\boldsymbol{x};\boldsymbol{\beta})$ that has the best fit to a series of data points $\{(\boldsymbol{x}_i, y_i)\}_i$.

- Linear model:

$$f(\boldsymbol{x};\boldsymbol{\beta}) = \boldsymbol{x}\boldsymbol{\beta}\,, \quad \boldsymbol{x} \in \mathbb{R}^{1\times D}\,, \boldsymbol{\beta} \in \mathbb{R}^{D\times 1}\,,$$
$$y_i = f(\boldsymbol{x})_i + \varepsilon_i\,, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2\mathbb{I})\,.$$



- Best fit:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}\,,$$

where

$$\mathbf{X} \equiv \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}, \quad \boldsymbol{y} \equiv \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}.$$

- Error estimation

$$\hat{\sigma}^2 = \sum_{i=1}^{N} \frac{[y_i - f(\boldsymbol{x}_i;\hat{\boldsymbol{\beta}})]^2}{N-D}\,,$$
$$\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2(\mathbf{X}^T\mathbf{X})^{-1}\,.$$

Today: nonlinear least-squares fitting

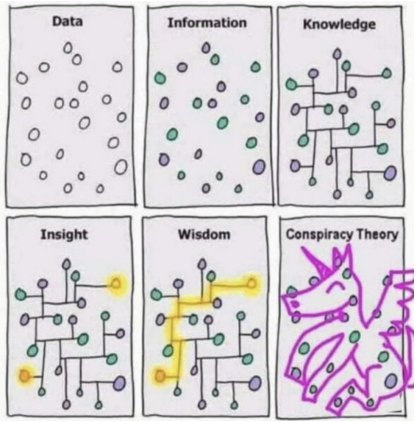
The process of constructing a parametrized nonlinear function $f(\boldsymbol{x};\boldsymbol{\beta})$ that has the best fit to a series of data points $\{(\boldsymbol{x}_i, y_i)\}_i$.

- Nonlinear model: $f(\boldsymbol{x};\boldsymbol{\beta})$ some generic nonlinear function with parameters $\boldsymbol{\beta}$.
  For example...
  - $f(t;\beta_1,\beta_2) = \beta_1 \mathrm{e}^{-\beta_2 t}$ (exponential decay).
  - $B_\lambda(\lambda;T) = \dfrac{2hc^2}{\lambda^5}\dfrac{1}{\mathrm{e}^{hc/(\lambda k_B T)}-1}$ (Plank's law for blackbody radiation).
- Assumption on errors (same as before):

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i\,, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})\,.$$

- Objective: minimize sum of squared residuals:

$$S_{\mathrm{res}} = \sum_i [y_i - f(\boldsymbol{x}_i)]^2\,.$$



Always be careful with overfitting...

Nonlinear least-squares fitting

We now try to minimize $S_{\rm res}$ in the same way as before:

$$S_{\rm res}(\beta) = \sum_i [y_i - f(\boldsymbol{x}_i; \beta)]^2 \,.$$

- and compute the partial derivatives...

$$\frac{\partial S_{\rm res}}{\partial \beta_j} = 2 \sum_i [f(\boldsymbol{x}_i; \beta) - y_i] \frac{\partial f(\boldsymbol{x}_i; \beta)}{\partial \beta_j} = \text{something nonlinear in } \boldsymbol{\beta} \text{ in general } \☹$$

- There is no closed-form solution so we have to proceed iteratively: starting from an initial guess for $\boldsymbol{\beta}$ and improve the guess little by little.
- Since we know how to solve linear least-squares, let's linearize f around the current value of $\boldsymbol{\beta}$:

$$f(\boldsymbol{x}; \boldsymbol{\beta} + \Delta\boldsymbol{\beta}) \simeq f(\boldsymbol{x}; \boldsymbol{\beta}) + \sum_j \frac{\partial f(\boldsymbol{x}; \boldsymbol{\beta})}{\partial \beta_j} \Delta\beta_j \,, \quad \text{for small } \Delta\boldsymbol{\beta} \,.$$

- The linearized problem becomes finding $\Delta\boldsymbol{\beta}$ such that

$$\sum_i \left[f(\boldsymbol{x}_i; \boldsymbol{\beta}) + \sum_j \frac{\partial f(\boldsymbol{x}_i; \boldsymbol{\beta})}{\partial \beta_j} \Delta\beta_j - y_i \right] \frac{\partial f(\boldsymbol{x}_i; \boldsymbol{\beta})}{\partial \beta_j} = 0 \,.$$

Nonlinear least-squares fitting

- The linearized problem becomes finding $\Delta\boldsymbol{\beta}$ such that

$$\sum_i \left[f(\boldsymbol{x}_i;\boldsymbol{\beta}) + \sum_j \frac{\partial f(\boldsymbol{x}_i;\boldsymbol{\beta})}{\partial \beta_j}\Delta\beta_j - y_i \right] \frac{\partial f(\boldsymbol{x}_i;\boldsymbol{\beta})}{\partial \beta_j} = 0\,.$$

- Let's clean up this equation a bit by defining:

$$r_i \equiv y_i - f(\boldsymbol{x}_i;\boldsymbol{\beta})\,, \quad J_{ij} = \frac{\partial f(\boldsymbol{x}_i;\boldsymbol{\beta})}{\partial \beta_j}\,,$$

- and after rearranging the terms, we get

$$\mathbf{J}^T\mathbf{J}\Delta\boldsymbol{\beta} = \mathbf{J}^T\boldsymbol{r} \quad \longrightarrow \quad \Delta\boldsymbol{\beta} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\boldsymbol{r}\,.$$

  This is a linear system that we know how to solve! ☺
  (Note that the *Jacobian* matrix **J** plays the role of **X** in the linear case.)
- Since the linearization of $f(\boldsymbol{x};\boldsymbol{\beta})$ only makes sense locally around $\boldsymbol{\beta}$, we move $\boldsymbol{\beta}$ by a **small** step in the direction of $\Delta\boldsymbol{\beta}$ for our next guess:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha\Delta\boldsymbol{\beta}\,,$$

  for some small $\alpha \in (0,1)$ (for example a constant $\alpha = 0.1$).
- We then iterate using the new guess until convergence.

Nonlinear least-squares fitting: summary

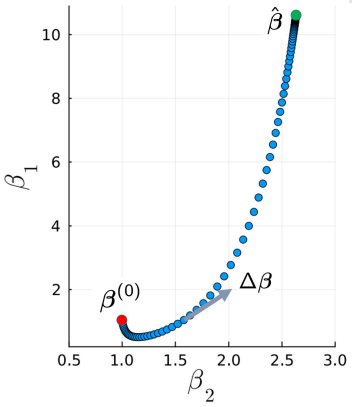
Algorithm: damped Gauss-Newton method

Input: dataset $\{(\boldsymbol{x}_i, y_i)\}_i$, model function $f(\boldsymbol{x}; \boldsymbol{\beta})$, small step length α, small threshold ϵ and a large `maxIter`.

1. Set $\boldsymbol{\beta}$ to some initial guess $\boldsymbol{\beta}^{(0)}$.
2. `DO k = 1, maxIter`
   - Construct the vector $\boldsymbol{r}$ of residuals $r_i \equiv y_i - f(\boldsymbol{x}_i; \boldsymbol{\beta})$.
   - Construct the Jacobian matrix $J_{ij} = \dfrac{\partial f(\boldsymbol{x}_i; \boldsymbol{\beta})}{\partial \beta_j}$.
   - Find the descent direction $\Delta\boldsymbol{\beta} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\boldsymbol{r}$.
   - `IF` $||\Delta\boldsymbol{\beta}||_2^2 \equiv \sum_j \Delta\beta_j^2 <= \epsilon$ `EXIT`.
   - Update fit parameters with $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha\Delta\boldsymbol{\beta}$.

   `END DO`

Output: Best fit parameters $\hat{\boldsymbol{\beta}}$.



Error estimation

- Estimator for the variance σ^2 of the error $\varepsilon_i = y_i - f(x_i)$:

$$\hat{\sigma}^2 = \sum_{i=1}^{N} \frac{\varepsilon_i^2}{N-D} = \frac{S_{\mathrm{res}}(\hat{\boldsymbol{\beta}})}{N-D}\,.$$

- Standard errors (SE) for the fit parameters:

$$\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2 \Big[\mathbf{J}(\widehat{\boldsymbol{\beta}})^T \mathbf{J}(\widehat{\boldsymbol{\beta}}) \Big]^{-1}, \quad \widehat{\mathrm{SE}}(\hat{\beta}_i) = \sqrt{\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})_{i,i}}\,.$$

  (Again, note that the *Jacobian* matrix **J** plays the role of **X** in the linear case.)

- What about R^2?
  ...It's not valid for nonlinear least-squares (even though some still use it).
  See, for example, "An evaluation of R^2 as an inadequate measure for nonlinear models in pharmacological and biochemical research: a Monte Carlo approach".

Assignment: estimate the sun surface temperature

Almost all the radiation that enters the Earth's atmosphere comes from the Sun, which originates in thermonuclear reactions in the core of the Sun. Most of the light emitted by the sun is characteristic of a blackbody radiator described by the Plank's law:

$$B_\lambda(\lambda; T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{hc/(\lambda k_B T)} - 1},$$

where T is the temperature and B_λ denotes the spectral energy density (intensity per unit wavelength) of the radiation. In the text file `sun_data.txt` you can find the data for B_λ (in $\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{nm}^{-1}$) as a function of λ (in microns) measured from sunlight. **Write a Fortran program to perform a nonlinear fit of the data and estimate the surface temperature of the sun.**

- Perform the fit with the nonlinear model $B(\lambda; \beta_1, \beta_2) = \dfrac{\beta_1}{\lambda^5 (e^{\beta_2/\lambda} - 1)}$
- Use the damped Gauss-Newton method and play with the hyperparameters. A good starting point could be an initial guess $\boldsymbol{\beta}^{(0)} = (1.0, 1.0)$, $\alpha = 0.1$, $\epsilon =$ `1.0E-6` and `maxIter=1000`.
- print the estimates for (β_1, β_2) and their standard errors in a text file `fit.txt`. Can you deduce the temperature from the fit parameters?
- **Bonus question**: Plot the measured spectrum and your fitted model using your favorite program (such as `gnuplot`) for you to appreciate your work.

Submit your code as `Ass10.YourLastName.f90` to `li.zejian@ictp.it` before the next lesson.

Help and hints for the assignment

- Write separate function for the model $B(\lambda;\beta)$ and for its partial derivatives.
- The partial derivatives are

$$\frac{\partial B(\lambda;\beta)}{\partial \beta_1} = \frac{1}{\lambda^5(\mathrm{e}^{\beta_2/\lambda}-1)}\,, \qquad \frac{\partial B(\lambda;\beta)}{\partial \beta_2} = -\frac{\beta_1 \mathrm{e}^{\beta_2/\lambda}}{\lambda^6(\mathrm{e}^{\beta_2/\lambda}-1)^2}\,.$$

- Include all functions in module(s).
- Useful built-in functions: `MATMUL` for matrix-matrix / matrix-vector multiplication, `TRANSPOSE` for transposing a matrix, `SUM` for summation...
- the text file has 2 lines of headers (to skip) and 1612 lines of numbers (to read).
- $hc/k_B \approx 14387.77$ K·micron. Divide this number by your fitted numerical value of β_2 [which is $hc/(k_B T)$] and you obtain the estimated temperature in K.
- If you use `gnuplot`:
  - If the file 'data.txt' has two columns t and x, then you can plot $x(t)$ with:
    `$ gnuplot -p -e "plot 'data.txt' using 1:2 with lines"`
  - If the file 'data.txt' has three columns t, x, and v, then you can plot $x(t)$ and $v(t)$ with:
    `$ gnuplot -p -e "plot for [col=2:3] 'data.txt' using 1:col with lines"`