

02610
Optimization and Data Fitting
Week 7: Nonlinear Least-Squares Problems

Yiqiu Dong

DTU Compute
Technical University of Denmark

Lecture Material

- J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd Edition, Springer.
 - ▶ Chapter 10: Least-squares problems.
 - ▶ We cover: 10.3.
- P. C. Hansen, V. Pereyra and G. Scherer, *Least Squares Data Fitting with Applications*, Johns Hopkins University Press.
 - ▶ Chapter 8: Nonlinear least squares problems.
 - ▶ We cover: 8.1 and 8.2.
 - ▶ Chapter 9: Algorithms for solving nonlinear LSQ problems.
 - ▶ We cover: 9.1, 9.2, and 9.3.

Nonlinearity

- A parameter α of the function appears **nonlinearly**, if the derivative $\partial f / \partial \alpha$ is a function of α .
- The model $\phi(x, t)$ is **nonlinear**, if at least one of the parameters in x appear nonlinearly.

Example: Consider the exponential decay model

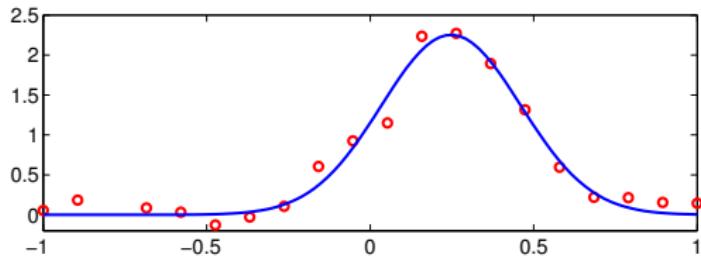
$$\phi(x_1, x_2, t) = x_1 e^{-x_2 t}.$$

We have:

- $\partial \phi / \partial x_1 = e^{-x_2 t}$ which is independent of x_1 ,
- $\partial \phi / \partial x_2 = -t x_1 e^{-x_2 t}$ which depends on x_2 .

Thus ϕ is a nonlinear model with the parameter x_2 appearing nonlinearly.

Fitting with a Gaussian Model



The non-normalized Gaussian function:

$$\phi(x, t) = x_1 e^{-(t-x_2)^2/(2x_3^2)}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

where x_1 is the amplitude, x_2 is the time shift, and x_3 determines the width of the Gaussian function.

The parameters x_2 and x_3 appear nonlinearly in this model.

Gaussian models also arise in many other data fitting problems.

Nonlinear Least Squares Problem

Find a parameter vector x^* which minimizes the sum-of-squares of the residuals:

$$\min_x f(x) \equiv \min_x \frac{1}{2} \|r(x)\|_2^2 = \min_x \frac{1}{2} \sum_{i=1}^m (y_i - \phi(x, t_i))^2,$$

where $x \in \mathbb{R}^n$ and, as usual,

$$r(x) = \begin{bmatrix} r_1(x) \\ \vdots \\ r_m(x) \end{bmatrix} = \begin{bmatrix} y_1 - \phi(x, t_1) \\ \vdots \\ y_m - \phi(x, t_m) \end{bmatrix} \in \mathbb{R}^m,$$

- Here y_i are the measured data corresponding to t_i .
- The nonlinearity arises **only** from $\phi(x, t)$.

The Jacobian and the Gradient of $f(\mathbf{x})$

The **Jacobian** $J(\mathbf{x})$ of the vector function $\mathbf{r}(\mathbf{x})$ is defined as the matrix with elements

$$[J(\mathbf{x})]_{ij} = \frac{\partial r_i(\mathbf{x})}{\partial x_j} = -\frac{\partial \phi(\mathbf{x}, t_i)}{\partial x_j}, \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

The i th row of $J(\mathbf{x})$ equals the transpose of the gradient of $r_i(\mathbf{x})$:

$$[J(\mathbf{x})]_{i,:} = \nabla r_i(\mathbf{x})^T = -\nabla \phi(\mathbf{x}, t_i)^T, \quad i = 1, \dots, m.$$

Thus the elements of the gradient of $f(\mathbf{x})$ are given by

$$[\nabla f(\mathbf{x})]_j = \frac{\partial f(\mathbf{x})}{\partial x_j} = \sum_{i=1}^m r_i(\mathbf{x}) \frac{\partial r_i(\mathbf{x})}{\partial x_j}$$

and it follows that the gradient is the vector

$$\boxed{\nabla f(\mathbf{x}) = J(\mathbf{x})^T \mathbf{r}(\mathbf{x})}.$$

The Hessian Matrix of $f(\mathbf{x})$

The elements of the **Hessian** of f , denoted $\nabla^2 f(\mathbf{x})$, are given by

$$[\nabla^2 f(\mathbf{x})]_{k\ell} = \frac{\partial^2 f(\mathbf{x})}{\partial x_k \partial x_\ell} = \sum_{i=1}^m \frac{\partial r_i(\mathbf{x})}{\partial x_k} \frac{\partial r_i(\mathbf{x})}{\partial x_\ell} + \sum_{i=1}^m r_i(\mathbf{x}) \frac{\partial^2 r_i(\mathbf{x})}{\partial x_k \partial x_\ell},$$

and it follows that the Hessian can be written as

$$\boxed{\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + \sum_{i=1}^m r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}),}$$

where

$$\begin{aligned} [\nabla^2 r_i(\mathbf{x})]_{k\ell} &= - [\nabla^2 \phi(\mathbf{x}, t_i)]_{k\ell} \\ &= - \frac{\partial^2 \phi(\mathbf{x}, t_i)}{\partial x_k \partial x_\ell}, \quad k, \ell = 1, \dots, n . \end{aligned}$$

The Optimality Conditions

The conditions for x^* to be a local minimizer of a twice continuously differentiable function f are

- **First-order optimality condition:**

$$\nabla f(x) = J(x)^T r(x) = 0 .$$

- **Second-order sufficient condition:**

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^m r_i(x) \nabla^2 r_i(x) \quad \text{is positive definite.}$$

Newton's Method

If $f(x)$ is twice continuously differentiable, we have the Newton iteration:
For $k = 0, 1, 2, \dots$

$$\begin{aligned}x_{k+1} &= x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \\&= x_k - \left(J(x_k)^T J(x_k) + S(x_k) \right)^{-1} J(x_k)^T r(x_k),\end{aligned}$$

where $S(x_k)$ denotes the matrix

$$S(x_k) = \sum_{i=1}^m r_i(x_k) \nabla^2 r_i(x_k).$$

Convergence. Locally quadratic convergence, but expensive – requires mn^2 derivatives to evaluate $S(x_k)$.

Newton's Method

If $f(x)$ is twice continuously differentiable, we have the Newton iteration:

For $k = 0, 1, 2, \dots$

$$\begin{aligned}x_{k+1} &= x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \\&= x_k - \left(J(x_k)^T J(x_k) + S(x_k) \right)^{-1} J(x_k)^T r(x_k),\end{aligned}$$

where $S(x_k)$ denotes the matrix

$$S(x_k) = \sum_{i=1}^m r_i(x_k) \nabla^2 r_i(x_k).$$

- The first term of the Hessian contains only $J(x)$, i.e., only first order derivatives!
- If the residuals are close to affine or are small near the solution, the second term will be very small.
- When the first term is dominant, one gets the main part of the Hessian “for free”.

The Gauss-Newton Method

If the residuals are close to affine or very small when near to the solution, a good alternative is to neglect the second term $S(x_k)$ of the Hessian. It turns out to the Gauss-Newton method, where $x_{k+1} = x_k + p_k^{\text{GN}}$ with p_k^{GN} obtained by solving

$$\left(J(x_k)^T J(x_k) \right) p_k^{\text{GN}} = -J(x_k)^T r(x_k).$$

If $J(x_k)$ has full rank

- The linear system of p_k^{GN} is actually the normal equations for the linear least squares problem

$$\min_{p_k^{\text{GN}}} \| J(x_k) p_k^{\text{GN}} - (-r(x_k)) \|_2^2.$$

- p_k^{GN} is a descent step.

Local Linear Approximation

If we introduce a Taylor expansion to approximate the nonlinear LSQ problem, then

$$\begin{aligned}\min_{p_k} f(x_k + p_k) &= \min_{p_k} \frac{1}{2} \|r(x_k + p_k)\|_2^2 \\ &\approx \min_{p_k} \frac{1}{2} \|r(x_k) + J(x_k)p_k\|_2^2.\end{aligned}$$

It is the same as the Gauss-Newton step in the full-rank case. Hence, the Gauss-Newton method is solving a linear approximation of the original nonlinear LSQ problem.

Damped Gauss-Newton Method

Implementations of the G-N method usually perform a line search in the direction p_k^{GN} , e.g., requiring the step length α_k to satisfy the Armijo condition:

$$\begin{aligned} f(x_k + \alpha_k p_k^{\text{GN}}) &< f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k^{\text{GN}} \\ &= f(x_k) + c_1 \alpha_k r(x_k)^T J(x_k) p_k^{\text{GN}}, \end{aligned}$$

with a constant $c_1 \in (0, 1)$.

This ensures that the reduction is (at least) proportional to both the parameter α_k and the directional derivative $\nabla f(x_k)^T p_k^{\text{GN}}$.

Line search make the algorithm (often) globally convergent.

Convergence. Can be quadratic if the neglected term in the Hessian is small. Otherwise it is linear.

Gauss-Newton Algorithm

Set the starting point x_0

loop

Solve

$$\min_p \|J(x_k) p + r(x_k)\|_2^2$$

to obtain the search direction p_k^{GN} ;

Choose a step length α_k so that there is enough descent;

Calculate the new iterate: $x_{k+1} = x_k + \alpha_k p_k^{GN}$;

Check for convergence;

end loop

Output x_{k+1} .

Example

We fit the one-parameter model $\phi(x, t) = e^{xt}$ to the data set

$$(t_1, y_1) = (1, 2), \quad (t_2, y_2) = (2, 4), \quad (t_3, y_3) = (3, y),$$

where y can take one of the four values 8, 3, -1, -8. The least squares problem is, for every y , to determine the single parameter x by minimizing

$$f(x) = \frac{1}{2} \sum_{i=1}^3 r_i(x)^2 = \frac{1}{2} [(e^x - 2)^2 + (e^{2x} - 4)^2 + (e^{3x} - y)^2].$$

Then, the two terms in the Hessian are

$$J(x)^\top J(x) = \sum_{i=1}^3 (t_i e^{xt_i})^2 \quad S(x) = \sum_{i=1}^3 r_i(x) t_i^2 e^{xt_i}.$$

Example

y	x^*	$f(x^*)$	$J(x^*)^T J(x^*)$	$S(x^*)$
8	0.6932	0	644.00	0
3	0.4401	1.639	151.83	9.0527
-1	0.0447	6.977	17.6492	8.3527
-8	-0.7915	41.145	0.4520	2.9605

y	x_0	Newton		Gauss-Newton	
		# iter.	rate	# iter.	rate
8	1	7	quadratic	5	quadratic
	0.6	6	quadratic	4	quadratic
3	1	9	quadratic	12	linear
	0.5	5	quadratic	9	linear
-1	1	10	quadratic	34	linear (slow)
	0	4	quadratic	32	linear (slow)
-8	1	12	quadratic	no convergence	
	-0.7	4	quadratic	no convergence	

The Levenberg-Marquardt Method

Similar to Gauss-Newton method, except that we replace the line search with a trust-region strategy where the norm of the step is bounded by the trust-region radius Δ_k .

$$\min \|J(x_k) p + r(x_k)\|_2^2 \quad \text{subject to} \quad \|p\|_2 \leq \Delta_k.$$

We can handle such an inequality constraint via the use of a Lagrange multiplier, and it turns out the step in Levenberg-Marquardt's method:

$$p_k^{\text{LM}} = \operatorname{argmin}_p \left\{ \|J(x_k) p + r(x_k)\|_2^2 + \lambda_k \|p\|_2^2 \right\}$$

where $\lambda_k > 0$ is a so-called *Lagrange parameter* for the constraint at the k th iteration.

The Levenberg-Marquardt Method

The optimization problem in each iteration

$$\min_p \left\{ \|J(x_k)p + r(x_k)\|_2^2 + \lambda_k \|p\|_2^2 \right\}$$

is equivalent to two forms:

$$\min_p \left\| \begin{bmatrix} J(x_k) \\ \sqrt{\lambda_k} I \end{bmatrix} p - \begin{bmatrix} -r(x_k) \\ 0 \end{bmatrix} \right\|_2^2$$

and

$$(J(x_k)^\top J(x_k) + \lambda_k I) p = -J(x_k)^\top r(x_k).$$

- For all $\lambda_k > 0$, p_k is a descent direction.
- This method is more robust in case of an ill-conditioned Jacobian, and can also handle a rank-deficient Jacobian.

Levenberg-Marquardt Algorithm

Set the starting point x_0

loop

Choose the Lagrange parameter λ_k ;

Solve the linear LSQ problem

$$\min_p \left\| \begin{bmatrix} J(x_k) \\ \sqrt{\lambda_k} I \end{bmatrix} p - \begin{bmatrix} -r(x_k) \\ 0 \end{bmatrix} \right\|_2^2$$

to obtain the step p_k^{LM} .

Calculate the new iterate: $x_{k+1} = x_k + p_k^{\text{LM}}$;

Check for convergence;

end loop

Output x_{k+1} .

Note: there is no line search (i.e., no α_k -parameter), its role is taken over by the Lagrange parameter λ_k .

The Role of Lagrange Parameter

Consider the Levenberg-Marquardt step, which we formally write as:

$$p_k^{\text{LM}} = - \left(J(x_k)^T J(x_k) + \lambda_k I \right)^{-1} J(x_k)^T r(x_k).$$

The parameter λ_k influences both the direction and the length of the step.

- If λ_k is close to 0, then p_k^{LM} tends to the Gauss-Newton step.
- If λ_k is very large, then p_k^{LM} is a short step approximately in the steepest descent direction.

How to Choose Lagrange Parameter

A strategy developed by Marquardt. The underlying principles are:

- ① The initial value $\lambda_0 \approx \|J(x_0)^T J(x_0)\|_2$.
- ② For subsequent steps, an improvement ratio is defined as:

$$\rho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(x_k) - f(x_{k+1})}{\frac{1}{2}(p_k^{\text{LM}})^T (\lambda_k p_k^{\text{LM}} - J(x_k)^T r(x_k))}.$$

Here, the denominator is the reduction in f predicted by the local linear model.

- If ρ_k is large, then the pure Gauss-Newton model is good enough, so λ_{k+1} can be made smaller than at the previous step.
- If ρ_k is small (or even negative), then a short steepest descent step should be used, i.e., λ_{k+1} should be increased.

Marquardt's Parameter Updating

Algorithm

```
if  $\rho_k > 0.75$  then
    Set  $\lambda_{k+1} = \lambda_k/3$ ;
else if  $\rho_k < 0.25$  then
    Set  $\lambda_{k+1} = 2\lambda_k$ ;
else
    Use  $\lambda_{k+1} = \lambda_k$ ;
end if

if  $\rho_k > 0$  then
    Accept the update:  $x_{k+1} = x_k + p_k^{\text{LM}}$ ;
else
    Reject the update:  $x_{k+1} = x_k$ ;
end if
```

Convergence. The Levenberg-Marquardt method is (often) globally convergent. It can reach quadratic convergent rate, if the neglected term in the Hessian is small. Otherwise, it is linear.

Example

We fit the one-parameter model $M(x, t) = e^{xt}$ to the data set

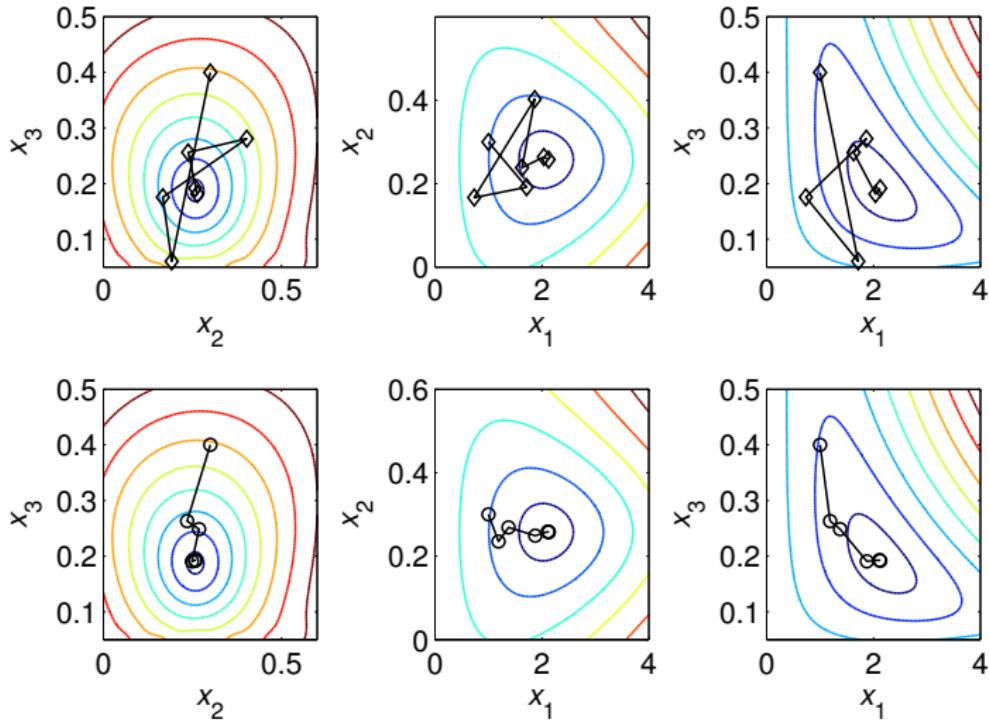
$$(t_1, y_1) = (1, 2), \quad (t_2, y_2) = (2, 4), \quad (t_3, y_3) = (3, y).$$

The least squares problem is minimizing

$$f(x) = \frac{1}{2} \sum_{i=1}^3 r_i(x)^2 = \frac{1}{2} [(e^x - 2)^2 + (e^{2x} - 4)^2 + (e^{3x} - y)^2].$$

		Gauss-Newton		Levenberg-Marquardt	
y	x_0	# iter.	rate	# iter.	rate
8	1	5	quadratic	10	quadratic
	0.6	4	quadratic	7	quadratic
3	1	12	linear	13	linear
	0.5	9	linear	10	linear
-1	1	34	linear (slow)	26	linear (slow)
	0	32	linear (slow)	24	linear (slow)
-8	1	no convergence		125	linear (very slow)
	-0.7	no convergence		120	linear (very slow)

G-N no line search (top) vs. L-M (bottom)



Conclusion

Characteristics	Newton	G-N	L-M
Ill-conditioned Jacobian	yes	yes (but slow)	yes
Rank-deficient Jacobian	yes	no	yes
Convergence $S(\mathbf{x}_k) = 0$	quadratic	quadratic	quadratic
Convergence $S(\mathbf{x}_k)$ small	quadratic	linear	linear
Convergence $S(\mathbf{x}_k)$ large	quadratic	slow or none	slow or none

scipy.optimize.least_squares

```
least_squares(fun, x0, jac='2-point', bounds=(-inf, inf), method='trf',
    ftol=1e-08, xtol=1e-08, gtol=1e-08, x_scale=1.0, loss='linear',
    f_scale=1.0, diff_step=None, tr_solver=None, tr_options={},
    jac_sparsity=None, max_nfev=None, verbose=0, args=(), kwargs={}) [source]
```

It solves LSQ problem

$$\min_{\|x\| \leq u} \frac{1}{2} \|f(x)\|_2^2 = \min_{\|x\| \leq u} \frac{1}{2} (f_1(x)^2 + \cdots + f_m(x)^2)$$

Inputs:

- fun: a function to compute the vector of residuals

$$f(x) = [f_1(x), \dots, f_m(x)]^T$$

- x0: an starting point;
- bounds: lower and upper bounds on x, i.e. bounds=(l, u);
- method: algorithm to perform minimization, e.g. 'trf', 'dogbox', 'lm', etc.