WIKIPEDIA

Levenberg–Marquardt algorithm

In mathematics and computing, the **Levenberg–Marquardt algorithm** (**LMA** or just **LM**), also known as the **Damped least-squares** (**DLS**) method, is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting.

The LMA is used in many software applications for solving generic curve-fitting problems. However, as with many fitting algorithms, the LMA finds only a local minimum, which is not necessarily the global minimum. The LMA interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent. The LMA is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum. For well-behaved functions and reasonable starting parameters, the LMA tends to be a bit slower than the GNA. LMA can also be viewed as Gauss–Newton using a trust region approach.

The algorithm was first published in 1944 by Kenneth Levenberg,[1] while working at the Frankford Army Arsenal. It was rediscovered in 1963 by Donald Marquardt,[2] who worked as a statistician at DuPont, and independently by Girard,[3] Wynne[4] and Morrison.[5]

Contents

The problem

The primary application of the Levenberg–Marquardt algorithm is in the least-squares curve fitting problem: given a set of m empirical pairs (x_i, y_i) of independent and dependent variables, find the parameters $\boldsymbol{\beta}$ of the model curve $f(x, \boldsymbol{\beta})$ so that the sum of the squares of the deviations $S(\boldsymbol{\beta})$ is minimized:

$$\hat{\boldsymbol{\beta}} \in \operatorname{argmin}_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) \equiv \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{m} [y_i - f(x_i, \boldsymbol{\beta})]^2,$$ which is assumed to be non-empty.

The solution

Like other numeric minimization algorithms, the Levenberg–Marquardt algorithm is an iterative procedure. To start a minimization, the user has to provide an initial guess for the parameter vector $\boldsymbol{\beta}$. In cases with only one minimum, an uninformed standard guess like $\boldsymbol{\beta}^{\mathrm{T}} = (1, 1, \ldots, 1)$ will work fine; in cases with multiple minima, the algorithm converges to the global minimum only if the initial guess is already somewhat close to the final solution.

In each iteration step, the parameter vector $\boldsymbol{\beta}$ is replaced by a new estimate $\boldsymbol{\beta} + \boldsymbol{\delta}$. To determine $\boldsymbol{\delta}$, the function $f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta})$ is approximated by its linearization:

$$f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\beta}) + \mathbf{J}_i \boldsymbol{\delta},$$

where

$$\mathbf{J}_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$

is the gradient (row-vector in this case) of f with respect to $\boldsymbol{\beta}$.

The sum $S(\boldsymbol{\beta})$ of square deviations has its minimum at a zero gradient with respect to $\boldsymbol{\beta}$. The above first-order approximation of $f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta})$ gives

$$S(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx \sum_{i=1}^{m} \left[y_i - f(x_i, \boldsymbol{\beta}) - \mathbf{J}_i \boldsymbol{\delta}\right]^2,$$

or in vector notation,

$$\begin{aligned} S(\boldsymbol{\beta} + \boldsymbol{\delta}) &\approx \left\|\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}\right\|^2 \\ &= \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}\right]^{\mathrm{T}} \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}\right] \\ &= \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right]^{\mathrm{T}} \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right] - \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right]^{\mathrm{T}} \mathbf{J}\boldsymbol{\delta} - (\mathbf{J}\boldsymbol{\delta})^{\mathrm{T}} \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right] + \boldsymbol{\delta}^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \mathbf{J} \boldsymbol{\delta} \\ &= \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right]^{\mathrm{T}} \left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right] - 2\left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right]^{\mathrm{T}} \mathbf{J}\boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \mathbf{J} \boldsymbol{\delta}. \end{aligned}$$

Taking the derivative of $S(\boldsymbol{\beta} + \boldsymbol{\delta})$ with respect to $\boldsymbol{\delta}$ and setting the result to zero gives

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathrm{T}}\left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right],$$

where $\mathbf{J}$ is the Jacobian matrix, whose i-th row equals $\mathbf{J}_i$, and where $\mathbf{f}(\boldsymbol{\beta})$ and $\mathbf{y}$ are vectors with i-th component $f(x_i, \boldsymbol{\beta})$ and y_i respectively. The above expression obtained for $\boldsymbol{\beta}$ comes under Gauss-Newton method. The Jacobian matrix as defined above is not (in general) a square matrix, but a rectangular matrix of size $m \times n$, where n is the number of parameters (size of the vector $\boldsymbol{\beta}$). The matrix multiplication $\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)$ yields the required $n \times n$ square matrix and the matrix-vector product on the right hand side yields a vector of size n. The result is a set n linear equations, which can be solved for $\boldsymbol{\delta}$.

Levenberg's contribution is to replace this equation by a "damped version":

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathrm{T}}\left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right],$$

where $\mathbf{I}$ is the identity matrix, giving as the increment $\boldsymbol{\delta}$ to the estimated parameter vector $\boldsymbol{\beta}$.

The (non-negative) damping factor λ is adjusted at each iteration. If reduction of S is rapid, a smaller value can be used, bringing the algorithm closer to the Gauss–Newton algorithm, whereas if an iteration gives insufficient reduction in the residual, λ can be increased, giving a step closer to the gradient-descent direction. Note that the gradient of S with respect to $\boldsymbol{\beta}$ equals $-2\left(\mathbf{J}^{\mathrm{T}}\left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right]\right)^{\mathrm{T}}$. Therefore, for large values of λ, the step will be taken approximately in the direction opposite to the gradient. If either the length of the calculated step $\boldsymbol{\delta}$ or the reduction of sum of squares from the latest parameter vector $\boldsymbol{\beta} + \boldsymbol{\delta}$ fall below predefined limits, iteration stops, and the last parameter vector $\boldsymbol{\beta}$ is considered to be the solution.

Levenberg's algorithm has the disadvantage that if the value of damping factor λ is large, inverting $\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}$ is not used at all. Fletcher provided the insight that we can scale each component of the gradient according to the curvature, so that there is larger movement along the directions where the gradient is smaller. This avoids slow convergence in the direction of small gradient. Therefore, Fletcher in his 1971 paper *A modified Marquardt subroutine for non-linear least squares* replaced the identity matrix $\mathbf{I}$ with the diagonal matrix consisting of the diagonal elements of $\mathbf{J}^{\mathrm{T}}\mathbf{J}$, thus making the solution scale invariant:

$$\left[\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda \operatorname{diag}\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)\right]\boldsymbol{\delta} = \mathbf{J}^{\mathrm{T}}\left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right].$$

A similar damping factor appears in Tikhonov regularization, which is used to solve linear ill-posed problems, as well as in ridge regression, an estimation technique in statistics.

Choice of damping parameter

Various more or less heuristic arguments have been put forward for the best choice for the damping parameter λ. Theoretical arguments exist showing why some of these choices guarantee local convergence of the algorithm; however, these choices can make the global convergence of the algorithm suffer from the undesirable properties of steepest descent, in particular, very slow convergence close to the optimum.

The absolute values of any choice depend on how well-scaled the initial problem is. Marquardt recommended starting with a value λ_0 and a factor $\nu > 1$. Initially setting $\lambda = \lambda_0$ and computing the residual sum of squares $S(\boldsymbol{\beta})$ after one step from the starting point with the damping factor of $\lambda = \lambda_0$ and secondly with λ_0/ν. If both of these are worse than the initial point, then the damping is increased by successive multiplication by ν until a better point is found with a new damping factor of $\lambda_0\nu^k$ for some k.

If use of the damping factor λ/ν results in a reduction in squared residual, then this is taken as the new value of λ (and the new optimum location is taken as that obtained with this damping factor) and the process continues; if using λ/ν resulted in a worse residual, but using λ resulted in a better residual, then λ is left unchanged and the new optimum is taken as the value obtained with λ as damping factor.

An effective strategy for the control of the damping parameter, called *delayed gratification*, consists of increasing the parameter by a small amount for each uphill step, and decreasing by a large amount for each downhill step. The idea behind this strategy is to avoid moving downhill too fast in the beginning of optimization, therefore restricting the steps available in future iterations and therefore slowing down convergence.[6] An increase by a factor of 2 and a decrease by a factor of 3 has been shown to be effective in most cases, while for large problems more extreme values can work better, with an increase by a factor of 1.5 and a decrease by a factor of 5.[7]

Geodesic acceleration

When interpreting the Levenberg–Marquardt step as the velocity $\boldsymbol{v}_k$ along a geodesic path in the parameter space, it is possible to improve the method by adding a second order term that accounts for the acceleration $\boldsymbol{a}_k$ along the geodesic

$$\boldsymbol{v}_k + \frac{1}{2}\boldsymbol{a}_k$$

where $\boldsymbol{a}_k$ is the solution of

$$\boldsymbol{J}_k\boldsymbol{a}_k = -\boldsymbol{f}_{vv}.$$

Since this geodesic acceleration term depends only on the directional derivative $f_{vv} = \sum_{\mu\nu} v_\mu v_\nu \partial_\mu \partial_\nu \boldsymbol{f}(\boldsymbol{x})$ along the direction of the velocity $\boldsymbol{v}$, it does not require to compute the full second order derivative matrix, requiring only a small overhead in terms of computing cost.[8] Since the second order derivative can be a fairly complex expression, it can be convenient to replace it with a finite difference approximation

$$\begin{aligned} f_{vv}^i &\approx \frac{\boldsymbol{f}_i(\boldsymbol{x} + h\boldsymbol{\delta}) - 2\boldsymbol{f}_i(\boldsymbol{x}) + \boldsymbol{f}_i(\boldsymbol{x} - h\boldsymbol{\delta})}{h^2} \\ &= \frac{2}{h}\left(\frac{\boldsymbol{f}_i(\boldsymbol{x} + h\boldsymbol{\delta}) - \boldsymbol{f}_i(\boldsymbol{x})}{h} - \boldsymbol{J}_i\boldsymbol{\delta}\right) \end{aligned}$$

where $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{J}$ have already been computed by the algorithm, therefore requiring only one additional function evaluation to compute $\boldsymbol{f}(\boldsymbol{x} + h\boldsymbol{\delta})$. The choice of the finite difference step h can affect the stability of the algorithm, and a value of around 0.1 is usually reasonable in general.[7]

Since the acceleration may point in opposite direction to the velocity, to prevent it to stall the method in case the damping is too small, an additional criterion on the acceleration is added in order to accept a step, requiring that

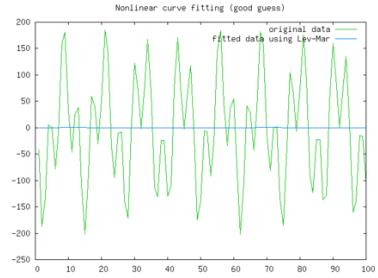
$$\frac{2\|\boldsymbol{a}_k\|}{\|\boldsymbol{v}_k\|} \leq \alpha$$

where α is usually fixed to a value lesser than 1, with smaller values for harder problems.[7]

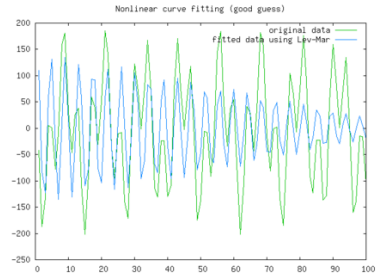
The addition of a geodesic acceleration term can allow significant increase in convergence speed and it is especially useful when the algorithm is moving through narrow canyons in the landscape of the objective function, where the allowed steps are smaller and the higher accuracy due to the second order term gives significative improvements.[7]

Example

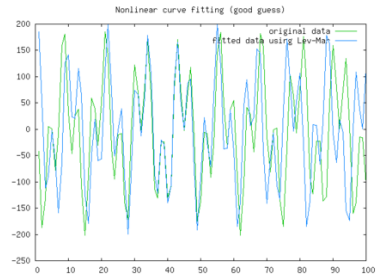
In this example we try to fit the function $y = a\cos(bX) + b\sin(aX)$ using the Levenberg–Marquardt algorithm implemented in GNU Octave as the *leasqr* function. The graphs show progressively better fitting for the parameters $a = 100$, $b = 102$ used in the initial curve. Only when the parameters in the last graph are chosen closest to the original, are the curves fitting exactly. This equation is an example of very sensitive initial conditions for the Levenberg–Marquardt algorithm. One reason for this sensitivity is the existence of multiple minima — the function $\cos(\beta x)$ has minima at parameter value $\hat{\beta}$ and $\hat{\beta} + 2n\pi$.



Poor fit



Better fit



Best fit

See also

- Trust region
- Nelder–Mead method
- Variants of the Levenberg–Marquardt algorithm have also been used for solving nonlinear systems of equations.[9]

References

1. Levenberg, Kenneth (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares". *Quarterly of Applied Mathematics*. **2** (2): 164–168. doi:10.1090/qam/10666 (https://doi.org/10.1090%2Fqam%2F10666).
2. Marquardt, Donald (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". *SIAM Journal on Applied Mathematics*. **11** (2): 431–441. doi:10.1137/0111030 (https://doi.org/10.1137%2F0111030). hdl:10338.dmlcz/104299 (https://hdl.handle.net/10338.dmlcz%2F104299).
3. Girard, André (1958). "Excerpt from *Revue d'optique théorique et instrumentale*". *Rev. Opt.* **37**: 225–241, 397–424.
4. Wynne, C. G. (1959). "Lens Designing by Electronic Digital Computer: I". *Proc. Phys. Soc. Lond.* **73** (5): 777–787. Bibcode:1959PPS....73..777W (https://ui.adsabs.harvard.edu/abs/1959PPS....73..777W). doi:10.1088/0370-1328/73/5/310 (https://doi.org/10.1088%2F0370-1328%2F73%2F5%2F310).
5. Morrison, David D. (1960). "Methods for nonlinear least squares problems and convergence proofs". *Proceedings of the Jet Propulsion Laboratory Seminar on Tracking Programs and Orbit Determination*: 1–9.
6. Transtrum, Mark K; Machta, Benjamin B; Sethna, James P (2011). "Geometry of nonlinear least squares with applications to sloppy models and optimization". *Physical Review E*. APS. **83** (3): 036701.
7. Transtrum, Mark K; Sethna, James P (2012). "Improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization". arXiv:1201.5885 (https://arxiv.org/abs/1201.5885).
8. "Nonlinear Least-Squares Fitting" (https://web.archive.org/web/20200414204913/https://www.gnu.org/software/gsl/doc/html/nls.html). GNU Scientific Library. Archived from the original (https://www.gnu.org/software/gsl/doc/html/nls.html) on 2020-04-14.
9. Kanzow, Christian; Yamashita, Nobuo; Fukushima, Masao (2004). "Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints". *Journal of Computational and Applied Mathematics*. **172** (2): 375–397. doi:10.1016/j.cam.2004.02.013 (https://doi.org/10.1016%2Fj.cam.2004.02.013).

Further reading

- Moré, Jorge J.; Sorensen, Daniel C. (1983). "Computing a Trust-Region Step" (https://digital.library.unt.edu/ark:/67531/metadc283525/m2/1/high_res_d/metadc283525.pdf) (PDF). *SIAM J. Sci. Stat. Comput.* **4** (3): 553–572. doi:10.1137/0904038 (https://doi.org/10.1137%2F0904038).