A Brief Description of the Levenberg-Marquardt Algorithm Implemened by levmar

Manolis Lourakis
Foundation for Research and Technology - Hellas

A Brief Description of the Levenberg-Marquardt Algorithm Implemened by `levmar`

Manolis I. A. Lourakis
Institute of Computer Science
Foundation for Research and Technology - Hellas (FORTH)
Vassilika Vouton, P.O. Box 1385, GR 711 10
Heraklion, Crete, GREECE

February 11, 2005

Abstract

The Levenberg-Marquardt (LM) algorithm is an iterative technique that locates the minimum of a function that is expressed as the sum of squares of nonlinear functions. It has become a standard technique for nonlinear least-squares problems and can be thought of as a combination of steepest descent and the Gauss-Newton method. This document briefly describes the mathematics behind `levmar`, a free LM C/C++ implementation that can be found at `http://www.ics.forth.gr/~lourakis/levmar`.

Introduction

The Levenberg-Marquardt (LM) algorithm is an iterative technique that locates the minimum of a multivariate function that is expressed as the sum of squares of non-linear real-valued functions [4, 6]. It has become a standard technique for non-linear least-squares problems [7], widely adopted in a broad spectrum of disciplines. LM can be thought of as a combination of steepest descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method. Next, a short description of the LM algorithm based on the material in [5] is supplied. Note, however, that a detailed analysis of the LM algorithm is beyond the scope of this report and the interested reader is referred to [5, 8, 9, 2, 10] for more comprehensive treatments.

The Levenberg-Marquardt Algorithm

In the following, vectors and arrays appear in boldface and ^{T} is used to denote transposition. Also, $||.||$ and $||.||_\infty$ denote the 2 and infinity norms respectively. Let f be an assumed functional relation which maps a *parameter vector* $\mathbf{p} \in \mathcal{R}^m$ to an estimated *measurement vector* $\hat{\mathbf{x}} = f(\mathbf{p}),\ \hat{\mathbf{x}} \in \mathcal{R}^n$. An initial parameter estimate $\mathbf{p}_0$ and a measured vector $\mathbf{x}$ are provided and it is desired to find the vector $\mathbf{p}^+$ that best satisfies the functional relation f, i.e. minimizes the squared distance $\epsilon^T\epsilon$ with $\epsilon = \mathbf{x} - \hat{\mathbf{x}}$. The basis of the LM algorithm is a linear approximation to f in the neighborhood of $\mathbf{p}$. For a small $||\delta_\mathbf{p}||$, a Taylor series expansion leads to the approximation

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p}, \tag{1}$$

where $\mathbf{J}$ is the Jacobian matrix $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$. Like all non-linear optimization methods, LM is iterative: Initiated at the starting point $\mathbf{p}_0$, the method produces a series of vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots,$ that converge towards a local minimizer $\mathbf{p}^+$ for f. Hence, at each step, it is required to find the $\delta_\mathbf{p}$ that minimizes the quantity $||\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p})|| \approx ||\mathbf{x} - f(\mathbf{p})\ -\ \mathbf{J}\delta_\mathbf{p}|| = ||\epsilon - \mathbf{J}\delta_\mathbf{p}||$. The sought $\delta_\mathbf{p}$ is thus the solution to a linear least-squares problem: the minimum is attained when $\mathbf{J}\delta_\mathbf{p} - \epsilon$ is orthogonal to the column space of $\mathbf{J}$. This leads to $\mathbf{J}^T(\mathbf{J}\delta_\mathbf{p} - \epsilon) = \mathbf{0}$, which yields $\delta_\mathbf{p}$ as the solution of the so-called *normal equations* [1]:

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T\epsilon. \tag{2}$$

The matrix $\mathbf{J}^T\mathbf{J}$ in the left hand side of Eq. (2) is the approximate Hessian, i.e. an approximation to the matrix of second order derivatives. The LM method actually solves a slight variation of Eq. (2), known as the *augmented normal equations*

$$\mathbf{N}\delta_\mathbf{p} = \mathbf{J}^T\epsilon, \tag{3}$$

where the off-diagonal elements of $\mathbf{N}$ are identical to the corresponding elements of $\mathbf{J}^T\mathbf{J}$ and the diagonal elements are given by $\mathbf{N}_{ii} = \mu + \left[\mathbf{J}^T\mathbf{J}\right]_{ii}$ for some

$\mu > 0$. The strategy of altering the diagonal elements of $\mathbf{J}^T\mathbf{J}$ is called *damping* and μ is referred to as the *damping term*. If the updated parameter vector $\mathbf{p} + \delta_{\mathbf{p}}$ with $\delta_{\mathbf{p}}$ computed from Eq. (3) leads to a reduction in the error ϵ, the update is accepted and the process repeats with a decreased damping term. Otherwise, the damping term is increased, the augmented normal equations are solved again and the process iterates until a value of $\delta_{\mathbf{p}}$ that decreases error is found. The process of repeatedly solving Eq. (3) for different values of the damping term until an acceptable update to the parameter vector is found corresponds to one iteration of the LM algorithm.

In LM, the damping term is adjusted at each iteration to assure a reduction in the error ϵ. If the damping is set to a large value, matrix $\mathbf{N}$ in Eq. (3) is nearly diagonal and the LM update step $\delta_{\mathbf{p}}$ is near the steepest descent direction. Moreover, the magnitude of $\delta_{\mathbf{p}}$ is reduced in this case. Damping also handles situations where the Jacobian is rank deficient and $\mathbf{J}^T\mathbf{J}$ is therefore singular [3]. In this way, LM can defensively navigate a region of the parameter space in which the model is highly nonlinear. If the damping is small, the LM step approximates the exact quadratic step appropriate for a fully linear problem. LM is adaptive because it controls its own damping: it raises the damping if a step fails to reduce ϵ; otherwise it reduces the damping. In this way LM is capable to alternate between a slow descent approach when being far from the minimum and a fast convergence when being at the minimum's neighborhood [3]. The LM algorithm terminates when at least one of the following conditions is met:

- The magnitude of the gradient of $\epsilon^T\epsilon$, i.e. $\mathbf{J}^T\epsilon$ in the right hand side of Eq. (2), drops below a threshold ε_1
- The relative change in the magnitude of $\delta_{\mathbf{p}}$ drops below a threshold ε_2
- The error $\epsilon^T\epsilon$ drops below a threshold ε_3
- A maximum number of iterations k_{max} is completed

If a covariance matrix $\mathbf{\Sigma_x}$ for the measured vector $\mathbf{x}$ is available, it can be incorporated into the LM algorithm by minimizing the squared $\mathbf{\Sigma_x}^{-1}$-norm $\epsilon^T\mathbf{\Sigma_x}^{-1}\epsilon$ instead of the Euclidean $\epsilon^T\epsilon$. Accordingly, the minimum is found by solving a weighted least squares problem defined by the *weighted normal equations*

$$\mathbf{J}^T\mathbf{\Sigma_x}^{-1}\mathbf{J}\delta_{\mathbf{p}} = \mathbf{J}^T\mathbf{\Sigma_x}^{-1}\epsilon. \qquad (4)$$

The rest of the algorithm remains unchanged. The complete LM algorithm is shown in pseudocode in Fig. 1. It is derived by slight modification of algorithm

3.16 in page 27 of [5]; more details regarding the LM algorithm can be found there. Indicative values for the user-defined parameters are $\tau = 10^{-3}$, $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 10^{-15}$, $k_{max} = 100$. `levmar` is a free C/C++ implementation of this LM algorithm that can be found at `http://www.ics.forth.gr/~lourakis/levmar`.

References

[1] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[2] C.T. Kelley. *Iterative Methods for Optimization*. SIAM Press, Philadelphia, 1999.

[3] M. Lampton. Damping-Undamping Strategies for the Levenberg-Marquardt Nonlinear Least-Squares Method. *Computers in Physics Journal*, 11(1):110–115, Jan./Feb. 1997.

[4] K. Levenberg. A Method for the Solution of Certain Non-linear Problems in Least Squares. *Quarterly of Applied Mathematics*, 2(2):164–168, Jul. 1944.

[5] K. Madsen, H.B. Nielsen, and O. Tingleff. Methods for Non-Linear Least Squares Problems. Technical University of Denmark, 2004. Lecture notes, available at `http://www.imm.dtu.dk/courses/02611/nllsq.pdf`.

[6] D.W. Marquardt. An Algorithm for the Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal of Applied Mathematics*, 11(2):431–441, Jun. 1963.

[7] H.D. Mittelmann. The Least Squares Problem. [web page] `http://plato.asu.edu/topics/problems/nlolsq.html`, Jul. 2004. [Accessed on 4 Aug. 2004.].

[8] H.B. Nielsen. Damping Parameter in Marquardt's Method. Technical Report IMM-REP-1999-05, Technical University of Denmark, 1999. Available at `http://www.imm.dtu.dk/~hbn`.

[9] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, New York, 1999.

Input: A vector function $f : \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.
Output: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $||\mathbf{x} - f(\mathbf{p})||^{\mathbf{2}}$.
Algorithm:

```
$k := 0$; $\nu := 2$; $\mathbf{p} := \mathbf{p}_0$;
$\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_{\mathbf{p}}$;
stop:=($||\mathbf{g}||_\infty \leq \varepsilon_1$); $\mu := \tau * \max_{i=1,\ldots,m}(A_{ii})$;
while (not stop) and ($k < k_{max}$)
    $k := k + 1$;
    repeat
        Solve $(\mathbf{A} + \mu\mathbf{I})\delta_{\mathbf{p}} = \mathbf{g}$;
        if $(||\delta_{\mathbf{p}}|| \leq \varepsilon_2||\mathbf{p}||)$
            stop:=true;
        else
            $\mathbf{p}_{new} := \mathbf{p} + \delta_{\mathbf{p}}$;
            $\rho := (||\epsilon_{\mathbf{p}}||^2 - ||\mathbf{x} - f(\mathbf{p}_{new})||^2)/(\delta_{\mathbf{p}}^T(\mu\delta_{\mathbf{p}} + \mathbf{g}))$;
            if $\rho > 0$
                $\mathbf{p} = \mathbf{p}_{new}$;
                $\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_{\mathbf{p}}$;
                stop:=($||\mathbf{g}||_\infty \leq \varepsilon_1$) or ($||\epsilon_{\mathbf{p}}||^2 \leq \varepsilon_3$);
                $\mu := \mu * \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$; $\nu := 2$;
            else
                $\mu := \mu * \nu$; $\nu := 2 * \nu$;
            endif
        endif
    until ($\rho > 0$) or (stop)
endwhile
$\mathbf{p}^+ := \mathbf{p}$;
```

Figure 1: Levenberg-Marquardt non-linear least squares algorithm; see text and [5, 8] for details.

[10] W.H. Press, S.A. Teukolsky, A.W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, 1992.