Algorithms

Aryana Haghjoo

November 9, 2022

1 How Does the Levenberg–Marquardt Algorithm Works?

Levenberg-Marquadt is an algorithm to solve non-linear least squares problems. If we have a set of parameters m, n pairs of independent and dependant variables (x_i, y_i), and a model curve $f(x, m)$, we are searching for m such that following sum (Chi-Square) will be minimized.

$$S(m) = \sum_{i=1}^{n} \left[y_i - f\left(x_i, m\right)\right]^2 \tag{1}$$

This algorithm is iterative and on each step the parameters m will be replaced by $m + dm$. Thus, we aim to find dm. In order to do so, we approximate the $f(x, m + dm)$ by:

$$f\left(x_i, m + dm\right) \approx f\left(x_i, m\right) + \mathbf{J}_i dm \tag{2}$$

where

$$\mathbf{J}_i = \frac{\partial f\left(x_i, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}} \tag{3}$$

is the gradient (later Jacobian Matrix) which will be the basis for calculation of the covariance matrix. Plugging 2 in 1, we get:

$$S(m + dm) \approx \sum_{i=1}^{n} \left[y_i - f\left(x_i, m\right) - \mathbf{J}_i dm\right]^2 \tag{4}$$

We take the derivative of $S(m + dm)$ with respect to m, and set the results equal to zero. Therfore, we get:

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right) dm = \mathbf{J}^{\mathrm{T}}[\mathbf{y} - \mathbf{f}(m)] \tag{5}$$

The Jacobian matrix displayed above has shapes of $n \times m$ where n is the number of data points and m is the number of parameters. In order to find the covariance matrix, we calculate $\mathbf{J}^T\mathbf{J}$, which will have the shapes of $m \times m$.

The process of solving the above equation iteratively, is called the Newton's method. But in some cases, this method will not converge to find the minimum. The solution is to add a damping term to the left hand side, changing the equation to:

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right) dm = \mathbf{J}^{\mathrm{T}}[\mathbf{y} - \mathbf{f}(m)] \tag{6}$$

On each iteration, we compare the chi-square to the last step. If it is smaller, the λ is divided by a constant factor and vice versa and if it becomes bigger, the λ will be multiplied to become a larger value. For practical purposes, if λ becomes less than a constant small value, we set it equal to zero.

There still remains a question on how to check the convergence of this method. If we reach an iteration where our chi-square has decreased and the value of λ equals to zero, we compare the difference in chi-square to a constant arbitrary threshold value. If the difference is less than this number, we declare that the algorithm has converged and we have reached a set of parameters that give us the lowest possible chi-square compared to out y_i values.