Numerical Optimization

Lecture Notes #24

Nonlinear Least Squares — Orthogonal Distance Regression

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Fall 2018

Outline

Summary: Linear Least Squares

Our study of non-linear least squares problems started with a look at **linear least squares**, where each residual $r_j(\bar{\mathbf{x}})$ is linear, and the Jacobian therefore is constant. The objective of interest is

$$f(\bar{\mathbf{x}}) = \frac{1}{2}\|J\bar{\mathbf{x}} + \bar{\mathbf{r}}_0\|_2^2, \quad \bar{\mathbf{r}}_0 = \bar{\mathbf{r}}(0),$$

solving for the stationary point $\nabla f(\bar{\mathbf{x}}^*) = 0$ gives the **normal equations**

$$J^T J\bar{\mathbf{x}}^* = -J^T\bar{\mathbf{r}}_0.$$

We have three approaches to solving the normal equations for $\bar{\mathbf{x}}^*$ — in increasing order of **computational complexity** and **stability:**

(i) Cholesky factorization of $J^T J$,
(ii) QR-factorization of J, and
(iii) Singular Value Decomposition of J.

Summary: Nonlinear Least Squares

Problem: Nonlinear Least Squares

$$\bar{\mathbf{x}}^* = \underset{\bar{\mathbf{x}}\in\mathbb{R}^n}{\arg\min}\,[f(\bar{\mathbf{x}})] = \underset{\bar{\mathbf{x}}\in\mathbb{R}^n}{\arg\min}\left[\frac{1}{2}\sum_{j=1}^m r_j(\bar{\mathbf{x}})^2\right], \quad m \geq n,$$

where the **residuals** $r_j(\bar{\mathbf{x}})$ are of the form $r_j(\bar{\mathbf{x}}) = y_j - \Phi(\bar{\mathbf{x}};\, t_j)$. Here, y_j are the **measurements** taken at the **locations/times** t_j, and $\Phi(\bar{\mathbf{x}};\, t_j)$ is our **model**.

The key approximation for **the Hessian**

$$\nabla^2 \mathbf{f}(\bar{\mathbf{x}}) = J(\bar{\mathbf{x}})^T J(\bar{\mathbf{x}}) + \sum_{j=1}^m r_j(\bar{\mathbf{x}})\nabla^2 r_j(\bar{\mathbf{x}}) \approx \mathbf{J}(\bar{\mathbf{x}})^\mathbf{T}\mathbf{J}(\bar{\mathbf{x}}).$$

Summary: Nonlinear Least Squares

Line-search algorithm: **Gauss-Newton**, with the subproblem:

$$\left[J(\bar{\mathbf{x}}_k)^T J(\bar{\mathbf{x}}_k)\right] \bar{\mathbf{p}}_k^{\text{GN}} = -\nabla f(\bar{\mathbf{x}}_k).$$

Guaranteed descent direction, fast convergence (as long as the Hessian approximation holds up) **equivalence** to a linear least squares problem (used for efficient, stable solution).

Summary: Nonlinear Least Squares

Trust-region algorithm: **Levenberg-Marquardt**, with the subproblem:

$$\bar{\mathbf{p}}_k^{\text{LM}} = \underset{\bar{\mathbf{p}}\in\mathbb{R}^n}{\arg\min} \frac{1}{2}\|J(\bar{\mathbf{x}}_k)\bar{\mathbf{p}} + \bar{\mathbf{r}}_k\|_2^2, \quad \text{subject to } \|\bar{\mathbf{p}}\| \leq \Delta_k.$$

Slight advantage over Gauss-Newton (global convergence), same local convergence properties; also (locally) equivalent to a linear least squares problem.

Summary: Nonlinear Least Squares

Hybrid Algorithms:

- When implementing Gauss-Newton or Levenberg-Marquardt, we should implement a **safe-guard** for the **large residual case**, where the Hessian approximation fails.
- If, after some reasonable number of iterations, we realize that the residuals are **not** going to zero, then we are better off switching to a general-purpose algorithm for non-linear optimization, such as a quasi-Newton (BFGS), or Newton method.

Fixed Regressor Models vs. Errors-In-Variables Models

So far we have assumed that there are **no errors** in the variables describing **where** / **when** the measurements are made, *i.e.* in the data set $\{t_j, y_j\}$ where t_j denote times of measurement, and y_j the measured value, we have assumed that **t_j are exact**, and the measurement errors are in y_j.

Under this assumption, the discrepancies between the model and the measured data are

$$\epsilon_j = y_j - \Phi(\bar{\mathbf{x}};\, t_j), \quad i = 1, 2, \ldots, m.$$

Next, we will take a look at the situation where we take errors in t_j into account — these models are known as **errors-in-variables models**, and their solutions in the linear case are referred to as **total least squares optimization**, or in the non-linear case as **orthogonal distance regression.**

Least Squares vs. Orthogonal Distance Regression

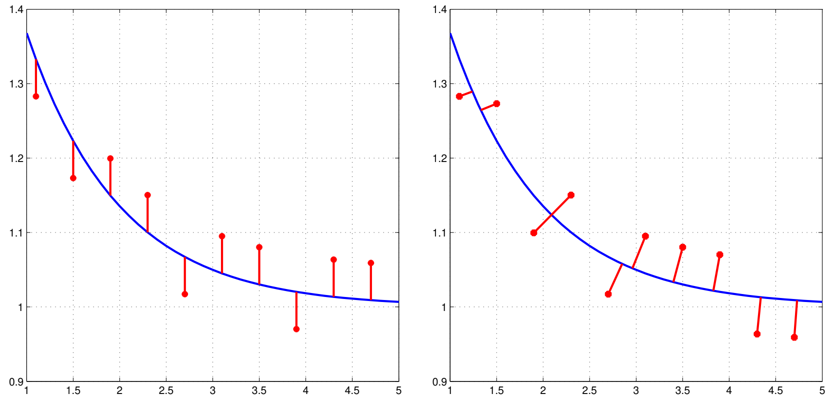

Figure: (left) An illustration of how the error is measured in standard (fixed regressor) least squares optimization. (right) An example of **orthogonal distance regression**, where we measure the shortest distance to the model curve. *[The right figure is actually not correct, why?]*

SAN DIEGO STATE UNIVERSITY

Orthogonal Distance Regression

For the mathematical formulation of orthogonal distance regression we introduce perturbations (errors) δ_j for the variables t_j, in addition to the errors ϵ_j for the y_j's.

We relate the measurements and the model in the following way

$$\epsilon_j = y_j - \Phi(\bar{\mathbf{x}};\ t_j + \delta_j),$$

and define the minimization problem:

$$(\bar{\mathbf{x}}^*, \bar{\delta}^*) = \arg\min_{\bar{\mathbf{x}},\bar{\delta}} \frac{1}{2} \sum_{j=1}^{m} \left[w_j^2 \left[y_j - \Phi(\bar{\mathbf{x}};\ t_j + \delta_j) \right]^2 + d_j^2 \delta_j^2 \right],$$

where $\bar{\mathbf{d}}$ and $\bar{\mathbf{w}}$ are two vectors of **weights** which denote the relative significance of the error terms.

SAN DIEGO STATE UNIVERSITY

Orthogonal Distance Regression: The Weights

The weight-vectors $\bar{\mathbf{d}}$ and $\bar{\mathbf{w}}$ must either be supplied by the modeler, or estimated in some clever way.

If all the weights are the same $w_j = d_j = \mathcal{C}$, then each term in the sum is simply the shortest distance between the point (t_j, y_j) and curve $\Phi(\bar{\mathbf{x}};\ t)$ (as illustrated in the previous figure).

In order to get the orthogonal-looking figure, I set $w_j = 1/0.5$ and $d_j = 1/4$, thus adjusting for the different scales in the t- and y-directions.

The shortest path between the point and the curve will be normal (orthogonal) to the curve at the point of intersection.

We can think of the scaling (weighting) as adjusting for measuring time in fortnights, seconds, milli-seconds, micro-seconds, or nano-seconds...

SAN DIEGO STATE UNIVERSITY

Orthogonal Distance Regression: In Terms of Residuals r_j

By identifying the $2m$ residuals

$$r_j(\bar{\mathbf{x}}, \bar{\delta}) = \begin{cases} w_j \left[y_j - \Phi(\bar{\mathbf{x}};\ t_j + \delta_j) \right] & j = 1, 2, \ldots, m \\ d_{j-m}\delta_{j-m} & j = (m+1), (m+2), \ldots, 2m \end{cases}$$

we can rewrite the optimization problem

$$(\bar{\mathbf{x}}^*, \bar{\delta}^*) = \arg\min_{\bar{\mathbf{x}},\bar{\delta}} \frac{1}{2} \sum_{i=1}^{m} w_j^2 \left[y_j - \Phi(\bar{\mathbf{x}};\ t_j + \delta_j) \right]^2 + d_j^2 \delta_j^2,$$

in terms of the $2m$-vector $\bar{\mathbf{r}}(\bar{\mathbf{x}}, \bar{\delta})$

$$(\bar{\mathbf{x}}^*, \bar{\delta}^*) = \arg\min_{\bar{\mathbf{x}},\bar{\delta}} \frac{1}{2} \sum_{i=1}^{2m} r_j(\bar{\mathbf{x}}, \bar{\delta})^2 = \arg\min_{\bar{\mathbf{x}},\bar{\delta}} \frac{1}{2} \|\mathbf{r}(\bar{\mathbf{x}}, \bar{\delta})\|_2^2.$$

SAN DIEGO STATE UNIVERSITY

Orthogonal Distance Regression → Least Squares

If we take a cold hard stare at the expression

$$(\bar{\mathbf{x}}^*, \bar{\delta}^*) = \arg\min_{\bar{\mathbf{x}},\bar{\delta}} \frac{1}{2}\sum_{i=1}^{2m} r_j(\bar{\mathbf{x}},\bar{\delta})^2 = \arg\min_{\bar{\mathbf{x}},\bar{\delta}} \frac{1}{2}\|\bar{\mathbf{r}}(\bar{\mathbf{x}},\bar{\delta})\|_2^2.$$

We realize that this is now a standard (nonlinear) least squares problem with $2m$ residuals and $(n+m)$ unknowns — $\{\bar{\mathbf{x}}, \bar{\delta}\}$.

We can use any of the techniques we have previously explored for the solution of the nonlinear least squares problem.

However, a straight-forward implementation of these strategies may prove to be quite expensive, since the number of parameters have doubled to $2m$ and the number of independent variables have grown from n to $(n+m)$. Recall that usually $m \gg n$, so this is a drastic growth of the problem.

Orthogonal Distance Regression → Least Squares: Problem Size

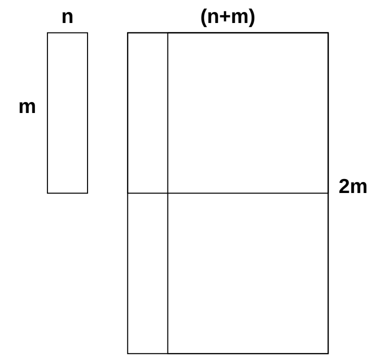

Figure: We recast ODR as a much larger standard nonlinear least squares problem.

Standard LSQ-solution via QR/SVD $\sim \mathcal{O}(mn^2)$, for $m \gg n$; slows down by a factor of $2(1+m/n)^2$.

ODR → Least Squares: Exploiting Structure

Fortunately we can save a lot of work by exploiting the structure of the Jacobian of the Least Squares problem originating from the orthogonal distance regression — **many entries are zero!**

$$\begin{aligned}
\frac{\partial r_j}{\partial \delta_i} &= w_j\frac{\partial[y_j - \Phi(\bar{\mathbf{x}};\, t_j+\delta_j)]}{\partial \delta_i} = 0, \quad \forall i,j \le m,\ i \ne j \\
\frac{\partial r_j}{\partial \delta_i} &= \frac{\partial[d_{j-m}\delta_{j-m}]}{\partial \delta_i} = \begin{cases} 0 & i \ne (j-m), j > m \\ d_{j-m} & i = (j-m),\ j > m \end{cases} \\
\frac{\partial r_j}{\partial x_i} &= \frac{\partial[d_{j-m}\delta_{j-m}]}{\partial x_i} = 0, \quad i = 1,2,\ldots,n, \quad j > m
\end{aligned}$$

Let $v_j = w_j\dfrac{\partial[y_j - \Phi(\bar{\mathbf{x}};\, t_j+\delta_j)]}{\partial \delta_j}$, and let $D = \mathrm{diag}(\bar{\mathbf{d}})$, and $V = \mathrm{diag}(\bar{\mathbf{v}})$, then we can write the Jacobian of the residual function in matrix form...

ODR → Least Squares: The Jacobian

We now have

$$J(\bar{\mathbf{x}},\bar{\delta}) = \left[\begin{array}{c|c} \widehat{J} & V \\ \hline 0 & D \end{array}\right],$$

where D and V are $m \times m$ diagonal matrices, $D = \mathrm{diag}(\bar{\mathbf{d}})$, and $V = \mathrm{diag}(\bar{\mathbf{v}})$, and $\widehat{J}$ is the $m \times n$ matrix defined by

$$\widehat{J} = \left[\frac{\partial[w_j(y_j - \Phi(\bar{\mathbf{x}};\, t_j+\delta_j))]}{\partial x_i}\right]_{\substack{j=1,2,\ldots,m \\ i=1,2,\ldots,n}}$$

We can now use this matrix in *e.g.* the Levenberg-Marquardt algorithm...

ODR → Least Squares: The Jacobian — Structure

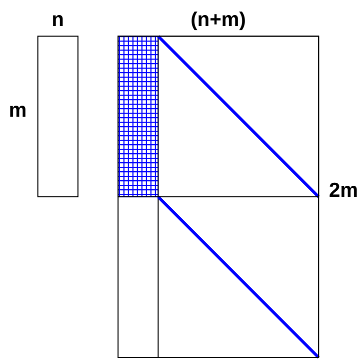

Figure: If we exploit the structure of the Jacobian, the problem is still somewhat tractable.

ODR → Least Squares → Levenberg-Marquardt 1 of 2

If we partition the step vector $\bar{\mathbf{p}}$, and the residual vector $\bar{\mathbf{r}}$ into

$$\bar{\mathbf{p}} = \begin{bmatrix} \bar{\mathbf{p}}_x \\ \bar{\mathbf{p}}_\delta \end{bmatrix}, \quad \bar{\mathbf{r}} = \begin{bmatrix} \tilde{\mathbf{r}}_1 \\ \tilde{\mathbf{r}}_2 \end{bmatrix}$$

where $\bar{\mathbf{p}}_x \in \mathbb{R}^n$, $\bar{\mathbf{p}}_\delta \in \mathbb{R}^m$, and $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \in \mathbb{R}^m$, then *e.g.* we can write the Levenberg-Marquardt subproblem in partitioned form

$$\left[\begin{array}{c|c} \widehat{J}^T\widehat{J} + \lambda I_n & \widehat{J}^T V \\ \hline V\widehat{J} & V^2 + D^2 + \lambda I_m \end{array}\right] \left[\begin{array}{c} \bar{\mathbf{p}}_x \\ \hline \bar{\mathbf{p}}_\delta \end{array}\right] = -\left[\begin{array}{c} \widehat{J}^T\tilde{\mathbf{r}}_1 \\ \hline V\tilde{\mathbf{r}}_1 + D\tilde{\mathbf{r}}_2 \end{array}\right]$$

Since the $(2,2)$-block $V^2 + D^2 + \lambda I_m$ is diagonal, we can eliminate the $\bar{\mathbf{p}}_\delta$ variables from the system...

ODR → Least Squares → Levenberg-Marquardt 2 of 2

$$\left[\begin{array}{c|c} \widehat{J}^T\widehat{J} + \lambda I_n & \widehat{J}^T V \\ \hline V\widehat{J} & V^2 + D^2 + \lambda I_m \end{array}\right] \left[\begin{array}{c} \bar{\mathbf{p}}_x \\ \hline \bar{\mathbf{p}}_\delta \end{array}\right] = -\left[\begin{array}{c} \widehat{J}^T\tilde{\mathbf{r}}_1 \\ \hline V\tilde{\mathbf{r}}_1 + D\tilde{\mathbf{r}}_2 \end{array}\right]$$

$$\bar{\mathbf{p}}_\delta = -\left[V^2 + D^2 + \lambda I_m\right]^{-1}\left[(V\tilde{\mathbf{r}}_1 + D\tilde{\mathbf{r}}_2) + V\widehat{J}\bar{\mathbf{p}}_x\right]$$

This leads to the $n \times n$-system $A\bar{\mathbf{p}}_x = \bar{\mathbf{b}}$, where

$$\begin{aligned} A &= \left[\widehat{J}^T\widehat{J} + \lambda I_n - \widehat{J}^T V\left[V^2 + D^2 + \lambda I_m\right]^{-1} V\widehat{J}\right] \\ \bar{\mathbf{b}} &= \left[-\widehat{J}^T\tilde{\mathbf{r}}_1 + \widehat{J}^T V\left[V^2 + D^2 + \lambda I_m\right]^{-1}\left[V\tilde{\mathbf{r}}_1 + D\tilde{\mathbf{r}}_2\right]\right]. \end{aligned}$$

Hence, the total cost of finding the LM-step is only marginally more expensive than for the standard least squares problem.

ODR → LSQ ($2m \times (n+m)$) → Levenberg-Marquardt → LSQ ($m \times n$)

The derived system *is typically very* ill-conditioned since we have formed a modified version of the normal equations $\widehat{J}^T\widehat{J} +$ *"stuff"*... With some work we can recast is as an $m \times n$ linear least squares problem $\bar{\mathbf{p}}_x = \arg\min_{\bar{\mathbf{p}}} \|\tilde{A}\bar{\mathbf{p}} - \tilde{\mathbf{b}}\|_2$, where

$$\begin{aligned} \tilde{A} &= \left[\widehat{J} + \lambda[\widehat{J}^T]^\dagger - V\left[V^2 + D^2 + \lambda I_m\right]^{-1} V\widehat{J}\right] \\ \tilde{\mathbf{b}} &= \left[-\tilde{\mathbf{r}}_1 + V\left[V^2 + D^2 + \lambda I_m\right]^{-1}\left[V\tilde{\mathbf{r}}_1 + D\tilde{\mathbf{r}}_2\right]\right] \end{aligned}$$

Where the "mystery factor" $[\widehat{J}^T]^\dagger$ is the **pseudo-inverse** of $\widehat{J}^T$. Expressed in terms of the QR-factorization $QR = \widehat{J}$, we have

$$\widehat{J}^T = R^T Q^T, \quad [\widehat{J}^T]^\dagger = QR^{-T},$$

Since $QR^{-T}R^TQ^T = I = R^TQ^TQR^{-T}$.

Software and References

`MINPACK`	Implements the Levenberg-Marquardt algorithm. Available for free from `http://www.netlib.org/minpack/`.
`ODRPACK`	Implements the orthogonal distance regression algorithm. Available for free from `http://www.netlib.org/odrpack/`.
Other	The NAG (Numerical Algorithms Group) library and HSL (formerly the Harwell Subroutine Library), implement several robust nonlinear least squares implementations.
GvL	Golub and van Loan's **Matrix Computations, 4th edition** (chapters 5–6) has a comprehensive discussion on orthogonalization and least squares; explaining in gory detail much of the linear algebra (*e.g.* the SVD and QR-factorization) we swept under the rug.

Summary
Orthogonal Distance Regression
Error Models
Weighted Least Squares / Orthogonal Distance Regression
ODR = Nonlinear Least Squares, Exploiting Structure

Index