Notes

Batch Estimation

Amro Al Baali

May 9, 2021

Contents

1 Why this document?

This document will be used as a guide to batch estimation on design variables that live in Lie groups. It'll be assumed that the random variables follow a Gaussian distribution. Therefore, the state estimation problem boils down to some least squares problem.

2 Linear least squares

Linear least squares is a special type of unconstrained optimization problem. It has the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^\mathsf{T}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right). \tag{1}$$

The error function is really *affince*, but that's how it is.

The objective function can be expanded to a quadratic form

$$\begin{aligned} J(\mathbf{x}) &= \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^\mathsf{T}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right) & (2)\\ &= \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}-\mathbf{b}^\mathsf{T}\mathbf{A}\mathbf{x}+\frac{1}{2}\mathbf{b}^\mathsf{T}\mathbf{b} & (3)\end{aligned}$$

which is a *convex* function in $\mathbf{x}$. Furthermore, the objective function is *strongly* quadratic function if $\mathbf{A}^\mathsf{T}\mathbf{A}$ is positive definite, which occurs if $\mathbf{A}$ has full *column rank*. [1]

[1] This assumption is usually valid since it is common to have more measurements than the number of design variables.

If $\mathbf{A}$ is full rank, then the optimization problem (1) has a unique minimizer and is given by solving the linear system

$$\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}^\star = \mathbf{A}^\mathsf{T}\mathbf{b}. \tag{4}$$

This equation is referred to as the *normal equations* [1].

There are efficient ways to solve (4) than inverting $\mathbf{A}^\mathsf{T}\mathbf{A}$. These include Cholesky and QR factorizations [2, 1].

3 Euclidean nonlinear least squares

Nonlinear least squares optimization problem is given by

$$\min_{\mathbf{x}\in\mathbf{R}^n} \frac{1}{2}\mathbf{e}(\mathbf{x})^\mathsf{T}\mathbf{e}(\mathbf{x}), \tag{5}$$

where $\mathbf{e}:\mathbb{R}^n\to\mathbb{R}^m$ is some nonlinear *error function*.

One way to solve this optimization iteratively is by linearizing the error function $\mathbf{e}(\mathbf{x})$ at some operating point $\bar{\mathbf{x}}$. This results in the Gauss-Newton equation [3, 1].

First, define the error on the optimizer. That is, let

$$\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}, \tag{6}$$

where $\bar{\mathbf{x}}$ is some operating point.[2] Plugging the error definition (6) into the error function gives

$$\mathbf{e}(\mathbf{x}) = \mathbf{e}(\bar{\mathbf{x}} + \delta\mathbf{x}) \tag{7}$$
$$\approx \mathbf{e}(\bar{\mathbf{x}}) + \mathbf{J}\delta\mathbf{x}, \tag{8}$$

where $\mathbf{J} \in \mathbb{R}^{m\times n}$ is the Jacobian of the error function $\mathbf{e}$ with respect to its design variables $\mathbf{x}$.

[2] The operating point will be updated at each iteration.

Plugging the perturbed error function (8) into the objective function gives

$$\tilde{J}(\delta\mathbf{x}) := J(\bar{\mathbf{x}} + \delta\mathbf{x}) \tag{9}$$
$$= \frac{1}{2}\mathbf{e}(\bar{\mathbf{x}} + \delta\mathbf{x})^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{x}} + \delta\mathbf{x}) \tag{10}$$
$$\approx \frac{1}{2}\left(\mathbf{e}(\bar{\mathbf{x}}) + \mathbf{J}\delta\mathbf{x}\right)^{\mathsf{T}}\left(\mathbf{e}(\bar{\mathbf{x}}) + \mathbf{J}\delta\mathbf{x}\right) \tag{11}$$
$$= \frac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\mathbf{J}^{\mathsf{T}}\mathbf{J}\delta\mathbf{x} + \mathbf{e}(\bar{\mathbf{x}})^{\mathsf{T}}\mathbf{J}\delta\mathbf{x} + \frac{1}{2}\mathbf{e}(\bar{\mathbf{x}})^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{x}}) \tag{12}$$

which is a quadratic[3] approximation of the objective function.

[3] Quadratic in $\delta\mathbf{x}$.

If $\mathbf{J}$ has full column rank, then $\tilde{J}(\delta\mathbf{x})$ is a strongly quadratic function. The minimizer of $\tilde{J}(\delta\mathbf{x})$ is then given by the solving the system of equations

$$\mathbf{J}^{\mathsf{T}}\mathbf{J}\delta\mathbf{x}^{\star} = -\mathbf{J}^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{x}}). \tag{13}$$

Again, this system of equations can be solved efficiently using Cholesky and QR factorizations.

Finally, using the error definition (8), the operating point can be updated using

$$\bar{\mathbf{x}}^{(j+1)} = \bar{\mathbf{x}}^{(j)} + \delta\mathbf{x}^{\star}, \tag{14}$$

where the superscript (j) is added to denote the Gauss-Newton iteration.

Gauss-Newton may perform poorly if the residual is large [4, 5], thus it's advisable to use line search methods when updating the operating point. That is, use

$$\bar{\mathbf{x}}^{(j+1)} = \bar{\mathbf{x}}^{(j)} + \alpha^{(j)}\delta\mathbf{x}^{\star}, \tag{15}$$

where $\alpha^{(j)}$ is a step-length computed using some heuristics like backtracking [4].

Another popular method is Levenberg-Marquardt [1, 4] which can be thought of as a damped version of Gauss-Newton.

4 Non-Euclidean nonlinear least squares

References

[1] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017. [Online]. Available: http://www.nowpublishers.com/article/Details/ROB-043

[2] G. H. Golub and C. F. Van Loan, *Matrix computations*, fourth edition ed., ser. Johns Hopkins studies in the mathematical sciences. Baltimore: The Johns Hopkins University Press, 2013, oCLC: ocn824733531.

[3] T. D. Barfoot, *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017. [Online]. Available: http://ebooks.cambridge.org/ref/id/CBO9781316671528

[4] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research. New York: Springer, 2006, oCLC: ocm68629100.

[5] R. R. Fletcher, *Practical methods of optimization*, 2nd ed. Chichester ;: Wiley, 1987, section: xiv, 436 pages : illustrations ; 24 cm.