Numerical Optimization

November 9, 2020

1 Introduction

In 3D computer vision, similarly to other estimation tasks, there are many minimization/maximization problems for which closed-form solutions do not exist. In this case, a vible solution to obtain the minimization is the application of numerical optimization techniques. They usually require an initial point, then try to make the cost function lower step-by-step.

2 Approximation by a Taylor serie

The cost function, denoted by $J(\boldsymbol{x})$, close to a given location $\boldsymbol{x}_0$ can be approximated by the well-known Taylor series technique:

$$J(\boldsymbol{x}_0 + \Delta\boldsymbol{x}) = J(\boldsymbol{x}_0) + \nabla J^T(\boldsymbol{x}_0)\Delta\boldsymbol{x} + \frac{1}{2}\Delta^T\boldsymbol{x}\boldsymbol{H}\Delta\boldsymbol{x} + \dots$$

where

$$\nabla J(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial J(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \frac{\partial J(\boldsymbol{x})}{\partial x_N} \end{bmatrix}$$

is the gradien of the cost function, and

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 J(\boldsymbol{x})}{\partial x_1^2} & \frac{\partial^2 J(\boldsymbol{x})}{\partial x_1 x_2} & \cdots & \frac{\partial^2 J(\boldsymbol{x})}{\partial x_1 x_N} \\ \frac{\partial^2 J(\boldsymbol{x})}{\partial x_2 x_1} & \frac{\partial^2 J(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 J(\boldsymbol{x})}{\partial x_2 x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\boldsymbol{x})}{\partial x_N x_1} & \frac{\partial^2 J(\boldsymbol{x})}{\partial x_N x_2} & \cdots & \frac{\partial^2 J(\boldsymbol{x})}{\partial x_N^2} \end{bmatrix}$$

is the so-called Hesse matrix. The latter one is symmetrix, because $\frac{\partial^2 J(\boldsymbol{x})}{\partial x_i x_j} = \frac{\partial^2 J(\boldsymbol{x})}{\partial x_j x_i}$ for all i and j.

For the one-parameter case, the Taylor serie is as follows:

$$f(x + \Delta x) = f(x_0) + f'(x_0)\Delta x + \frac{1}{2}f^{(2)}(x_0)\Delta^2 x + \dots$$

2.1 Gradient method

The simplest strategy is to move the value x_0 to the steepet descent. In this case,

$$\Delta x = -\alpha \nabla J$$

where α is a parameter to be set. Usually, it is empirically tuned.

2.2 Newton method

In the latter case of this document, only terms of the Taylor serie until the quadratic one is considered as it is written in the formulas above. (Higher terms are hidden by the dots.) If the initial value $\boldsymbol{x}_0$ is given, the optimal value for step $\Delta \boldsymbol{x}$ is given by derivating the cost function w.r.t. $\Delta \boldsymbol{x}$:

$$\frac{\partial J}{\partial \Delta \boldsymbol{x}} = \boldsymbol{\nabla} J + \boldsymbol{H} \Delta \boldsymbol{x} = 0$$

Then the step is calculated as

$$\Delta \boldsymbol{x} = -\boldsymbol{H}^{-1} \boldsymbol{\nabla} J$$

2.3 Gauss-Newton method

This method is designed for least-squares optimization. In this case, a cost function can usually be written by summing the square of the terms as

$$J = \sum_{j=1}^{M} f_j^2(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^T \boldsymbol{f}(\boldsymbol{x}),$$

where

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix}.$$

The gradient of the cost function is obtained as follows:

$$\nabla J = 2 \nabla^T f(\boldsymbol{x}) f(\boldsymbol{x}),$$

where $\nabla f(\boldsymbol{x})$ is a $M \times N$ matrix:

$$\nabla^T f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_M}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_N} & \frac{\partial f_2}{\partial x_N} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix},$$

since

$$\frac{\partial J}{\partial x_i} = \frac{\partial \sum_j f_j^2(\boldsymbol{x})}{\partial x_i} = 2 \sum f_j(\boldsymbol{x}) \frac{\partial f_j(\boldsymbol{x})}{\partial x_i}.$$

The second derivatives are as follows:

$$\frac{\partial^2 J}{\partial x_i \partial x_k} = 2 \sum_j \frac{\partial f_j(\boldsymbol{x})}{\partial x_i} \frac{\partial f_j(\boldsymbol{x})}{\partial x_k} + 2 \sum_j f_j(\boldsymbol{x}) \frac{\partial^2 f_j(\boldsymbol{x})}{\partial x_i \partial x_k}$$

If the second term is omitted,

$$\frac{\partial^2 J}{\partial x_i \partial x_k} \approx 2 \sum_j \frac{\partial f_j(\boldsymbol{x})}{\partial x_i} \frac{\partial f_j(\boldsymbol{x})}{\partial x_k}$$

Then the Hesse matrix is as follows:

$$H \approx 2 \nabla^T f(\boldsymbol{x}) \nabla f(\boldsymbol{x})$$

Then the iteration itself is as follows:

$$\Delta \boldsymbol{x} = -\boldsymbol{H}^{-1} \boldsymbol{\nabla} J = \left(\nabla^T f(\boldsymbol{x}) \nabla f(\boldsymbol{x}) \right)^{-1} \nabla^T f(\boldsymbol{x}) f(\boldsymbol{x})$$

3 Levenberg-Marquardt algorithm

The Levenberg Marquardt method is the mixture of gradient and Gauss-Newton algorithms. The update step is as the combination of gradient and Gauss-Newton method:

$$\Delta \boldsymbol{x} = \left(\nabla^T f(\boldsymbol{x}) \nabla f(\boldsymbol{x}) + \alpha I \right)^{-1} \nabla^T f(\boldsymbol{x}) f(\boldsymbol{x})$$

If $\alpha = 0$, Gauss-Newton method is obtained, if α is large, the gradient step dominates the update procedure.