Least-Squares Estimation of Nonlinear Parameters

Sasanka- 2019 & Viswanadh – 2019112011
International Insitute of Information Technology, Hyderabad, India

Overview

1. Introduction
2. Previous Algorithms
3. Levenberg-Marquardt Method.
4. Applications
5. Experimental Setup and Analysis
6. Conclusion

Introduction

- The model to be fitted to the data is defined as

$$E(y) = f(x_1, x_2, ..., x_m; b_1, b_2, ..., b_k) = f(\mathbf{x}, \mathbf{b}) \quad (1)$$

- where x1, x2, ..., xm are independent variables, b1, b2, ..., bk are the k parameters to estimate, E(y) is the expected value of the dependent variable y. To define the objective function, let the data points be denoted by
- The objective function to minimize to get the k parameters is given as

$$\Phi = \sum_{n=1}^{\infty} [Y_i - \hat{Y}_i]^2 = ||\mathbf{Y} - \hat{Y}||^2 (3)$$

- where $\hat{\mathbf{Y}}_i$ is the value of predicted y at the ith data point

Gauss-Newton Method

- The method is similar to expanding f in a Taylor series and in the vectorized form it can be written as $$\langle Y(\mathbf{X_i}, \mathbf{b} + \delta_t)\rangle = f(\mathbf{X_i}, \mathbf{b}) + \sum_{j=1}^{k} \left(\frac{\partial f_i}{\partial b_j}\right)(\delta_t)_j, \quad (4)$$
- The vector δt is a small correction to b, which can be found by least-squares method of setting $\frac{\partial(\Phi)}{\partial \delta_j} = 0$, for all j. This can be written as

$$A\boldsymbol{\delta_t} = \mathbf{g}, \quad (6)$$

where

$$A^{[k\times k]} = P^T P, \quad (7)$$

$$P^{[n\times k]} = \left(\frac{\partial f_i}{\partial b_j}\right) \quad (8)$$

where $i = 1, 2, 3, \ldots n; j = 1, 2, \ldots, k$

$$g^{[k\times 1]} = \left(\sum_{i=1}^{n} (Y_i - f_i)\frac{\partial f_i}{\partial b_j}\right) = P^T(\mathbf{Y} - \mathbf{f}_0) = \quad (9)$$

where $j = 1, 2, \ldots k$

Drawback of Gauss-Newton Method

- In practice it is found helpful to correct b by only a fraction of δt; otherwise the extrapolation may be beyond the region where f can be adequately represented by and would cause divergence of the iterates.
- So even though we have a good algorithm that predicts the values accurately, we always have to worry about the convergence of the process.

Gradient Descent Method

- The algorithm is implemented by correcting b by only a small value δg in the direction of the negative gradient of Φ. A step size $K\delta_g$, $0 < K \leq 1$ is used to control the amount of change in b after δ_g has specified the direction.

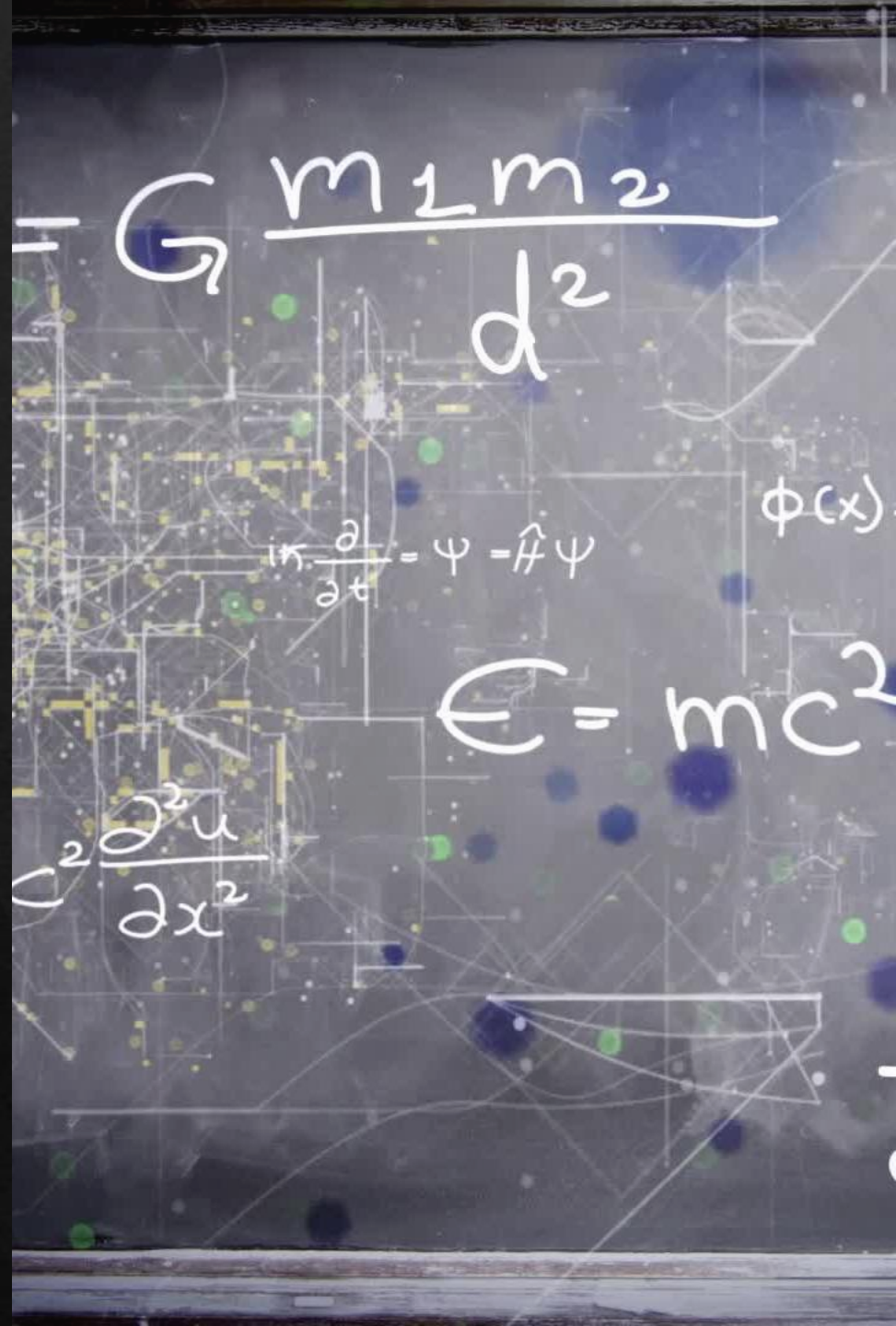
$$\boldsymbol{\delta_g} = -\left(\frac{\partial \Phi}{\partial b_1}, \frac{\partial \Phi}{\partial b_2}, \ldots, \frac{\partial \Phi}{\partial b_k}\right)^T$$

Drawback of Gradient Descent Method

- Various modified steepest-descent methods have been employed to compensate partially for the typically poor conditioning of the Φ surface which leads to very slow convergence of the gradient methods.
- So even though the values converge, the process is slow.

Motivation behind Levenberg-Marquardt method

- If we come up with a method that uses the convergence ability of gradient descent algorithm and quick&accurate convergence ability of the Gauss-Newton algorithm, then it would be an ideal algorithm.



Scale of Measurement

- We should scale the b-space in units of the standard deviations of the derivatives $\frac{\partial fi}{\partial b_j}$, taken over the sample points i = 1, 2, ..., n. We define a scaled matrix A* and a scaled vector g* :

$$A^* = a^*_{jj'} = \frac{a_{jj'}}{\sqrt{a_{jj}}\sqrt{a_{j'j'}}}$$

and

$$g^* = (g^*_j) = \left(\frac{g_j}{\sqrt{a_{jj}}}\right)$$

and solve for the Taylor series correction using

$$A^* \boldsymbol{\delta}^*_{\mathbf{t}} = g^*$$

Then

$$\delta_j = \frac{\delta^*_j}{\sqrt{a_{jj}}}$$

Levenberg-Marquardt Algorithm

- In order to minimize Φ locally, equation for r^{th} iteration can be constructed as

$$\left(A^{*(r)} + \lambda^{(r)} I\right) \boldsymbol{\delta}_1^{*(r)} = \mathbf{g}^{*(r)}$$

- Once above equation is solved for $\delta_1^{*(r)}$, $\delta_1^{(r)}$ can be obtained using normalisation. The new trial vector is given by $\mathbf{b}^{(r+1)} = \mathbf{b}^{(r)} + \boldsymbol{\delta}_1^{(r)}$
- It is necessary to select $\lambda^{(r)}$ such that

$$\Phi^{(r+1)} \leq \Phi^{(r)}$$

Steps

1) Let $v > 1$, and initialize $\lambda^{(0)} = 10^{-2}$.

2) Compute $\Phi\left(\lambda^{(r-1)}/v\right)$ and $\Phi\left(\lambda^{(r-1)}\right)$.
   i) If $\Phi\left(\lambda^{(r-1)}/v\right) \leq \Phi^{(r)}$, let $\lambda^{(r)} = \lambda^{(r-1)}/v$.
   ii) If $\Phi\left(\lambda^{(r-1)}/v\right) > \Phi^{(r)}$ and $\Phi\left(\lambda^{(r-1)}\right) \leq \Phi^{(r)}$, let $\lambda^{(r)} = \lambda^{(r-1)}$.
   iii) If $\Phi\left(\lambda^{(r-1)}/v\right) > \Phi^{(r)}$ and $\Phi\left(\lambda^{(r-1)}\right) > \Phi^{(r)}$, increase λ by successive multiplication by v until for some smallest w, $\Phi\left(\lambda^{(r-1)}v^{w}\right) \leq \Phi^{(r)}$

3) Set $\mathbf{b}^{(r)} \leftarrow \mathbf{b}^{(r-1)} + \delta_l$, where $\boldsymbol{\delta}_\mathbf{l}$ is obtained using equations (16) and (17) with $\lambda = \lambda^{(r)}$.

4) Set $r \leftarrow r + 1$.

5) Repeat steps 2 through 4.

Applications

Curve and Surface fitting

Given n data points p1, …, pn in R^2 or R^3. We need to fit a surface over these points.

The problem is reformulated as $\min_{a_0,\ldots,a_k} \frac{f^2(p_i; a_0, \ldots, a_k)}{\|\nabla f(p_i; a_0, \ldots, a_k)\|^2}$

Surface reconstruction

A robotic hand can reconstruct an unknown surface patch by touch. The idea is to track along three concurrent curves on the surface while collecting tactile data points (x_k , y_k , z_k)

The problem is reformulated as

$$\min_a f(a)$$

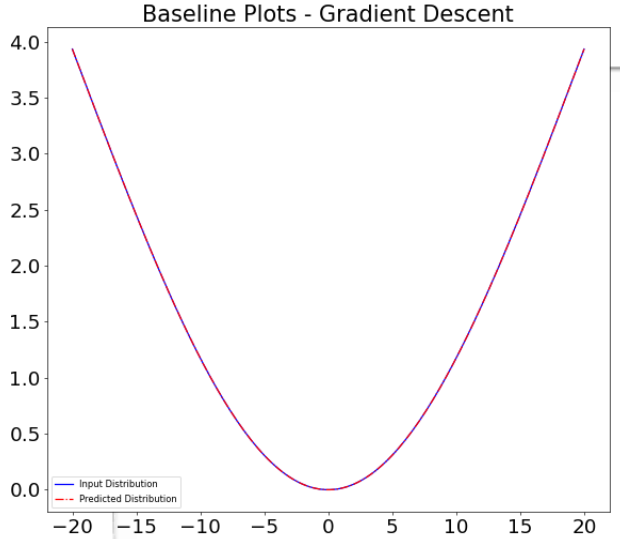
$$f(a) = \frac{1}{n}\sum_{k=1}^{n} (z(x_k, y_k) - z_k)^2 .$$

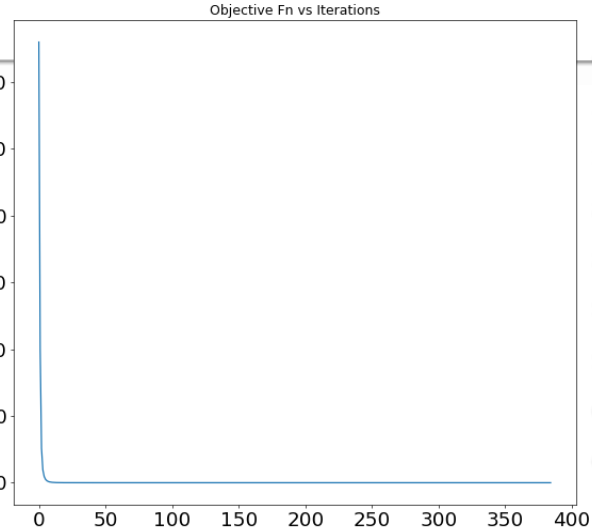
Implementation

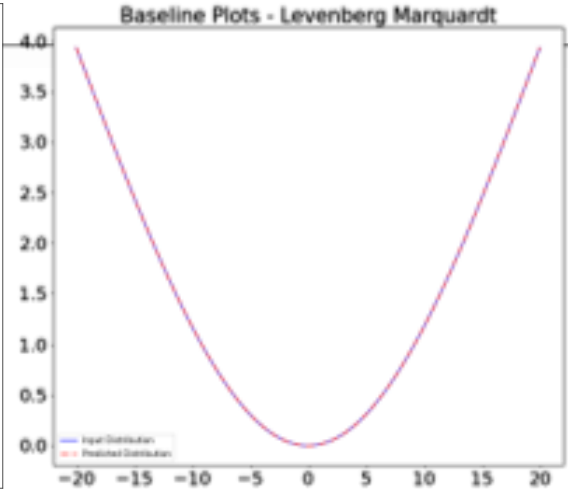
- We have taken the original function to be $f(x) = A * (1 - \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}})$

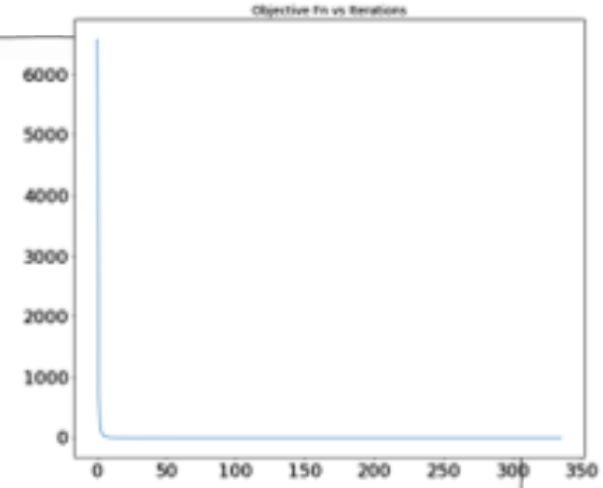
A = 10 is the scaling factor, μ = 0 is the mean and σ = 20 is the standard deviation. These three are the nonlinear parameters to be estimated.

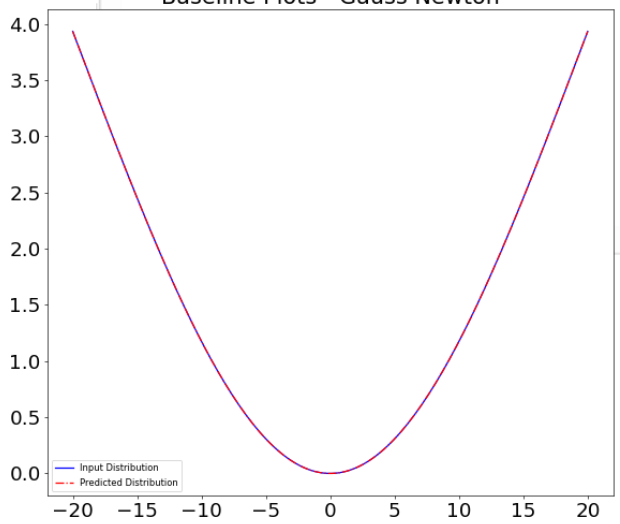
- Baseline: The experiment is performed for initial guess of [1, 1, 1], learning rate of 0.02 and 150 observation points.
- Experiment 1: In this experiment the initialization of the parameters is changed to [5, 10, 15].
- Experiment 2: The number of observation points are changed to 50.
- Experiment 3: A gaussian noise of μ=0.5 and σ=2 is added to the observations.

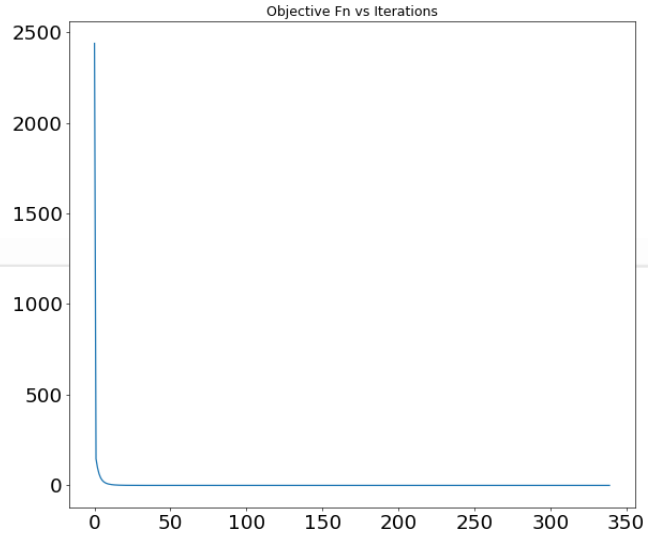

Baseline Plots - Gradient Descent
Input Distribution
Predicted Distribution


Objective Fn vs Iterations


Baseline Plots - Levenberg Marquardt


Objective Fn vs Iterations


Baseline Plots - Gauss Newton
Input Distribution
Predicted Distribution


Objective Fn vs Iterations

Results

Analysis

- We notice the number of iterations is roughly

  Gauss-Newton ≤ Levenberg-Marquardt ≤ Gradient Descent.

- The predicted values from Gradient descent is slightly bad. The predicted values from Gauss-Newton and Levenberg-Marquardt are almost the exact values taken as the original curve.
- So we can conclude that the new algorithm overcomes both the shortcomings and combines their good features.

Thank You