Project report: Gauss-Newton optimization method

Nicolas Munke Cilano
Vidar Gimbringer
Artis Vijups

Supervisor: Stefan Diehl

December 14, 2024

1 Introduction

In this project, we mainly investigate the problem of fitting the function

$$\varphi(\boldsymbol{x};t) = x_1 e^{-x_2 t} + x_3 e^{-x_4 t},$$

where $\boldsymbol{x} = (x_1, x_2, x_3, x_4)^T$ are parameters, to a given set of data points $(t_i, y_i), i = 1, \ldots, m$.

In particular, we seek to minimize the sum of the squared distances from each data point (t_i, y_i) to the point $(t_i, \varphi(\boldsymbol{x}; t_i))$. If we denote the distance by $r_i(\boldsymbol{x}) = \varphi(\boldsymbol{x}; t_i) - y_i$, then our goal is to

$$\underset{\boldsymbol{x}\in\mathbb{R}^4}{\text{minimize}}\ f(\boldsymbol{x}) \quad \text{where} \quad f(\boldsymbol{x}) = \sum_{i=1}^{m} r_i(\boldsymbol{x})^2.$$

To solve this minimization problem, we use the Gauss-Newton method.

2 Methods

2.1 Gauss-Newton method

Let $r(\boldsymbol{x}) = (r_1(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T$. The method is set up on the idea that

$$r(\boldsymbol{x} + \boldsymbol{\delta}) \approx r(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{\delta},$$

where $J(\boldsymbol{x})$ is the Jacobian matrix of $r(\boldsymbol{x})$ and $\boldsymbol{\delta}$ is an increment vector, also a direction vector.

This can be thought of as an extension to more dimensions of the strategy of moving along the tangent line to estimate a nearby value.

We then observe that

$$\begin{aligned} f(\boldsymbol{x} + \boldsymbol{\delta}) &= \sum_{i=1}^{m} r_i(\boldsymbol{x} + \boldsymbol{\delta})^2 \\ &= r(\boldsymbol{x} + \boldsymbol{\delta})^T r(\boldsymbol{x} + \boldsymbol{\delta}) \\ &\approx (r(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{\delta})^T (r(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{\delta}). \end{aligned}$$

Since we want the output of f to be the minimum possible, the gradient of this approximation should be 0. Writing that as an equation and simplifying, we end up with

$$J(\boldsymbol{x})^T J(\boldsymbol{x})\boldsymbol{\delta} = -J(\boldsymbol{x})^T r(\boldsymbol{x}).$$

This presents the following iterative algorithm:

1. Solve the linear system $J(\boldsymbol{x})^T J(\boldsymbol{x})\boldsymbol{\delta} = -J(\boldsymbol{x})^T r(\boldsymbol{x})$ for $\boldsymbol{\delta}$.
2. Determine an optimal step length λ for direction $\boldsymbol{\delta}$ using a line search algorithm.
3. Update $\boldsymbol{x}$ to $\boldsymbol{x} + \lambda\boldsymbol{\delta}$.

We repeat these actions until the step $\lambda\boldsymbol{\delta}$ is smaller than our chosen tolerance.

2.2 Line search algorithm

For the line search mentioned in the second step of the Gauss-Newton iteration algorithm, we use Armijo's rule on $F(\lambda) = f(\boldsymbol{x} + \lambda\boldsymbol{\delta})$.

Let $T(\lambda) = F(0) + \varepsilon F'(0)\lambda$ be a straight line through $(0, F(0))$ with less negative slope than the point's tangent, so $0 < \varepsilon < 1$.

Armijo's rule is made up of two (upper and lower) conditions, which are

$$F(\lambda) \leq T(\lambda) \quad \text{and} \quad F(\alpha\lambda) \geq T(\alpha\lambda) \text{ for fixed } \alpha > 1.$$

This rule ensures λ will be in an *interval* of points where F is substantially smaller. Computationally, this is faster than looking for a perfect choice for λ.

So that λ satisfies Armijo's rule, we choose it as follows:

1. Make an initial guess for λ.
2. Repeatedly scale λ up by α until it satisfies the lower condition.
3. Repeatedly scale λ down by α until it satisfies the upper condition.

3 Project work

3.1 Responsibilities

3.2 Structure

The project is stored on GitHub as a repository. It is made up of:

- phi1.m, phi2.m, data1.m, data2.m, grad.m as provided,
- gaussnewton.m, the implementation of the Gauss-Newton method,
- line_search.m, the line search algorithm based on Armijo's rule,
- script.m, the main script containing tests and tasks,
- report.tex and report.pdf, forming this report,
- metadata files.

3.3 Results

We fit the two functions from phi1.m and phi2.m,

$$\varphi_1(\boldsymbol{x};t) = x_1 e^{-x_2 t}, \quad \boldsymbol{x} = (x_1, x_2)^T$$

and

$$\varphi_2(\boldsymbol{x};t) = \varphi(\boldsymbol{x};t) = x_1 e^{-x_2 t} + x_3 e^{-x_4 t}, \quad \boldsymbol{x} = (x_1, x_2, x_3, x_4)^T,$$

to two sets of data points, `d1` and `d2` from data1.m and data2.m.

The results are:

Case	Optimal point $\boldsymbol{x}$	$\Vert\nabla f(\boldsymbol{x})\Vert_2$	$\Vert r(\boldsymbol{x})\Vert_\infty$
`d1`, φ_1	$(10.8108, 2.4786)^T$	$3.7450 \cdot 10^{-4}$	1.6287
`d2`, φ_1	$(12.9789, 1.7861)^T$	$8.0031 \cdot 10^{-2}$	1.0397
`d1`, φ_2	$(6.3445, 10.5866, 6.0959, 1.4003)^T$	$4.0651 \cdot 10^{-5}$	0.4334
`d2`, φ_2	$(4.1741, 0.8747, 9.7390, 2.9208)^T$	$9.5220 \cdot 10^{-3}$	0.1182

Appendix