

MECH 503 LAMMPS Project Report

Jincong Li
60539939

April 8, 2024

GN-m

Gauss-Newton method iteratively refines the parameter estimates to minimize the discrepancy between observed data and the model. This is particularly suited for problems where the model is a non-linear function of the parameters but linear with respect to the changes in the parameters.

Given a set of observations are the independent variables and β represents the parameters to be estimated, the goal is to minimize the sum of squared residuals.

The Gauss-Newton method updates the parameter estimates in each iteration using the formula:

where J is the Jacobian matrix of partial derivatives of the model functions with respect to the parameters β , and r is the vector of residuals between the observed and predicted values.

Marquardt's Modification introduces a damping factor λ to improve the convergence properties, especially when the Jacobian matrix

J is not invertible or near singular. The updated formula for parameter adjustment becomes:

Notice that the λ could be adjusted dynamically during iterations: when

λ is high, the algorithm behaves more like gradient descent, taking smaller steps; when

λ is low, it behaves more like the Gauss-Newton method, taking larger, more aggressive steps.

The essence of Marquardt's modification is to combine the strong points of both the Gauss-Newton method and gradient descent, allowing for a more versatile and robust approach to parameter estimation in non-linear models. The algorithm iteratively adjusts

λ and the parameters β until convergence is reached, typically when the change in the sum of squared residuals is below a certain threshold.

Covariance Matrix Calculation:

After obtaining the parameter estimates through the optimization process (e.g., Gauss-Newton method with Marquardt's modification), we calculate the covariance matrix of the parameter estimates. This matrix is derived from the inverse of the Hessian matrix, which, in the context of the Gauss-Newton method, is approximated by $J^T J$, where J is the Jacobian matrix of partial derivatives of the residuals with respect to the parameters. In cases where $J^T J$ is singular or near-singular (indicating potential issues with model identifiability or data collinearity), we used the pseudo-inverse for stability.

Standard Errors of the Parameter Estimates: The diagonal elements of the covariance matrix provide the variances of the parameter estimates. Taking the square root of these diagonal elements yields the standard errors of the estimates.

Confidence Interval Calculation: With the standard errors, we can compute the confidence intervals for the parameters. For a given confidence level (e.g., 95%), we determine the critical value from the t-distribution, considering the degrees of freedom (which typically depend on the number of data points and the number of parameters). The confidence interval for each parameter is then calculated as:

NMA

The Nelder-Mead algorithm is a popular direct search method used for minimizing an objective function in a multidimensional space. Unlike the Gauss-Newton or Levenberg-Marquardt algorithms, the Nelder-Mead method does not require the objective function to be

differentiable, making it particularly useful for optimizing non-linear and non-smooth functions. It's commonly applied in parameter estimation tasks for algebraic models where gradient-based methods might struggle due to discontinuities or the absence of derivatives.

How the Nelder-Mead Algorithm Works The algorithm operates using a simplex, which in two dimensions is a triangle. The simplex represents a set of candidate solutions that are iteratively updated to explore the space of parameters to find the minimum of the objective function.

The method involves the following key steps:

Initialization: Start with an initial simplex, which is typically generated by providing a starting point and then adding points that are slightly offset in the direction of each dimension.

Evaluation: Calculate the objective function value (e.g., the error or cost) at each vertex of the simplex.

Transformation: Update the simplex through a series of transformations: reflection, expansion, contraction, and shrinkage, based on the comparison of the objective function values at the vertices.

Reflection: Reflects the worst point (the point with the highest objective function value) across the centroid of the remaining points, possibly replacing the worst point if the reflected point is better.

Expansion: If the reflection is successful and finds a new best point, the algorithm tries to take a larger step in the same direction to see if it can improve further.

Contraction: If reflection does not result in improvement, the algorithm performs a contraction to move the worst point closer to the centroid, hoping to find a better position within a tighter search space.

Shrinkage: If neither reflection nor contraction significantly improves the worst point's position, the algorithm reduces the size of the simplex around the best point, shrinking the search space in hopes of finer exploration.

Termination: These steps are repeated until a termination criterion is met, which could be a small enough change in the objective function value, a sufficiently small simplex size, or a maximum number of iterations. The Nelder-Mead method does not require derivatives of the objective function, making it useful for optimization problems where derivatives are difficult or impossible to calculate. However, it's not guaranteed to find the global minimum, especially in problems with multiple local minima.

GN-m

GN-m

bootstrap method

So since in the direct search method, the hessian matrix or the derivatives are not available, the typical computational process for

covariance does not work, so we need another method.

The bootstrap method is a powerful statistical tool used to estimate the distribution of a statistic (like the mean or variance) from a set of data. It involves repeatedly sampling from the data set with replacement to create "bootstrap samples," and then computing the statistic of interest for each sample. This generates a distribution of the statistic, from which confidence intervals and other properties can be derived. The method is especially useful when the theoretical distribution of the statistic is unknown or difficult to derive.

Key Steps in the Bootstrap Method: **Resampling:** From your original data set of size N , randomly sample N observations with replacement to create a bootstrap sample. This sample is the same size as the original but may contain duplicates.

Computation: Calculate the statistic of interest (e.g., mean, variance, parameter estimates) for the bootstrap sample.

Repetition: Repeat steps 1 and 2 many times (e.g., 1000 or more iterations) to create a distribution of the statistic.

Estimation: Use the distribution obtained from step 3 to estimate the confidence intervals or variance of the statistic.

Confidence Interval Calculation: The confidence intervals are then calculated from these distributions. For a 95% confidence interval, the 2.5th and 97.5th percentiles of the estimated values for each parameter are used.

Practical Considerations: **Number of Iterations:** More iterations

increase the accuracy of the confidence interval estimates but also require more computational resources.

Random Sampling with Replacement: Ensures that each bootstrap sample is an independent and identically distributed sample of the data, mirroring the sampling variability in the original dataset.

Applicability: While bootstrap is widely applicable, its validity depends on the data being representative of the population. In cases where the original sample may not represent the population well, bootstrap estimates may be biased.

This method's strength lies in its simplicity and applicability to a wide range of problems, especially in situations where traditional parametric inference is challenging.