LEAST-SQUARES OPTIMISATION

Machine Learning

IIIy CSE IDP and IDDMP

Refer : Stephen Marsland

Motivation

- **Least-squares Optimisation is Widely Used in Modelling**
  - Appears in regression, parameter fitting, signal processing, computer vision
  - Natural choice when measurement noise is Gaussian
- **Mathematical Convenience**
  - Quadratic error function yields smooth, differentiable landscape
  - Analytical properties (convexity in linear case)
- **Computational Efficiency**
  - Exploits linear algebra structure (e.g. normal equations, QR decomposition)
  - Specialized solvers scale to very large data sets
- **Foundation for More Complex Methods**
  - Basis for iterative nonlinear methods (e.g. Levenberg–Marquardt)
  - Serves as subproblem in trust-region and Gauss–Newton algorithms

What Is Least-Squares Optimisation?

- **Goal:** Find parameters **x** that make model predictions $f(\mathbf{x}; t_i)$ as close as possible to observed values y_i .
- **Residual:** For each data point **i**, error is $r_i(\mathbf{x}) = y_i - f(\mathbf{x}; t_i)$
- **Objective:** Minimise the total squared error

$$f(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|_2^2$$

- **Special Structure:**

  If f is **linear** in **x**, you can solve for x directly (normal equations).

  If f is **nonlinear**, you use iterative methods (e.g, Levenberg–Marquardt).

- **Why "Least Squares"?**

Squaring residuals penalises large errors and yields smooth, well-behaved optimisation problems.

The Levenberg–Marquardt Algorithm

- **Objective Function**
  We frame least-squares fitting as minimising half the squared norm of the residual vector,

$$f(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2$$

  where r(x) collects all individual errors.

- **The Jacobian** J(x) contains all first-order partial derivatives of each residual with respect to each parameter.
  It tells us how a small change in parameters will affect each residual.

- **Gradient and Hessian**
  - The **gradient** of the objective is simply

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^\top \mathbf{r}(\mathbf{x})$$

  - The **Hessian** combines a term $J^\top J$ (like Gauss–Newton) with additional curvature from second derivatives of individual residuals.

Transpose of the Jacobian of **r**

$$\mathbf{J}^T(\mathbf{x}) = \left\{ \begin{array}{cccc} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_2}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_1} \\ \frac{\partial r_1}{\partial x_2} & \frac{\partial r_2}{\partial x_2} & \cdots & \frac{\partial r_m}{\partial x_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial r_1}{\partial x_n} & \frac{\partial r_2}{\partial x_n} & \cdots & \frac{\partial r_m}{\partial x_n} \end{array} \right\} = \left[\frac{\partial r_j}{\partial x_i} \right]_{j=1,\ldots,m,\ i=1,\ldots,n}$$

Exploiting Least-Squares Structure

Cheap Hessian Information

The **Jacobian J** alone gives the dominant Hessian term $\mathbf{J}^T\mathbf{J}$ with essentially no extra work.

Linear Residuals → Quadratic Objective

When each residual depends linearly on the parameters, the objective is exactly quadratic and second derivatives of residuals drop out.

Normal Equations

A second-order Taylor e›

$$\mathbf{J}^T\mathbf{J}\mathbf{x} = -\mathbf{J}^T\mathbf{r}(\mathbf{x})$$ Equation 1

Direct Solution

We can solve this as

$$\mathbf{x} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}$$ Equation 2

Solving with SVD in Linear Least-Squares

Decompose the Normal Matrix

Compute the singular value decomposition of

$$J^T J = USV^T$$

Equation 3

where **U** and **V** are orthogonal matrices and **S** is a diagonal matrix.

Computing the Solution

Substituting **equation 3** into **equation 2** we get :

$$x = VSU^T J^T r$$

which can be rearranged (using properties of transposes) into the compact SVD form.

Levenberg–Marquardt Algorithm (Trust-Region Approach)

Model Approximation

Replace full Hessian by Gauss–Newton approximation:

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

Neglect second-derivative (residual) terms for efficiency

Trust-Region Subproblem

At iteration **k**, solve $\min_{\mathbf{p}} \frac{1}{2}\|\mathbf{J}_k\mathbf{p} + \mathbf{r}_k\|_2^2, \ \ \|\mathbf{p}\| \leq \Delta_k$

Δ_k : trust-region radius (how far the linear model is trusted).

Damping (Levenberg–Marquardt Step)

Introduce damping parameter $\nu \geq 0$ to enforce trust-region:

$$(\mathbf{J}^T\mathbf{J} + \nu\mathbf{I})\mathbf{p} = -\mathbf{J}^T\mathbf{r}$$

Equivalently, enlarge ν if step is poor, reduce if step is good.

The Levenberg–Marquardt Algorithm

The Levenberg–Marquardt Algorithm

- Given start point $\mathbf{x}_0$
- While $\mathbf{J}^T\mathbf{r}(\mathbf{x})$ >tolerance and maximum number of iterations not exceeded:
  - repeat
    - solve $(\mathbf{J}^T\mathbf{J} + \nu\mathbf{I})\mathbf{dx} = -\mathbf{J}^T\mathbf{r}$ for $\mathbf{dx}$ using linear least-squares
    - set $\mathbf{x}_{\text{new}} = \mathbf{x} + \mathbf{dx}$
    - compute the ratio of the actual and prediction reductions:
      - actual $= \|f(\mathbf{x}) - f(\mathbf{x}_{new})\|$
      - predicted $= \nabla f^T(\mathbf{x}) \times \mathbf{x}_{new} - \mathbf{x}$
      - ρ = actual/predicted
    - if $0 < \rho < 0.25$:
      - accept step: $\mathbf{x} = \mathbf{x}_{\text{new}}$
    - else if $\rho > 0.25$:
      - accept step: $\mathbf{x} = \mathbf{x}_{\text{new}}$
      - increase trust region size (reduce ν)
    - else:
      - reject step
      - reduce trust region (increase ν)
  - until $\mathbf{x}$ is updated or maximum number of iterations is exceeded

Summary

Motivation for Least-Squares

- Ideal for models with squared, Gaussian-distributed errors
- Produces smooth, convex objectives in linear settings and manageable landscapes in nonlinear scenarios

Core Benefits

- **Analytical Solutions**: Direct methods (normal equations or SVD) deliver exact parameter estimates for linear problems
- **Efficient Iterative Solvers:** Gauss–Newton and Levenberg–Marquardt exploit Jacobian structure to accelerate convergence

Levenberg–Marquardt Highlights

- **Trust-Region Framework**: Integrates curvature information with gradient-descent stability
- **Simplified Hessian**: Employs the Gauss–Newton approximation, omitting expensive residual second derivatives
- **Subproblem Formulation**: Restricts each update to a region where the linear model is reliable
- **Adaptive Damping**: Dynamically adjusts the damping factor—expanding the trust region when steps succeed and contracting it when they do not

Key Takeaway

Least-squares optimisation combines robust linear-algebraic solutions with powerful nonlinear algorithms, making it a cornerstone technique in data fitting, machine learning, and scientific computing.