13. Nonlinear least squares

- definition and examples
- derivatives and optimality condition
- Gauss–Newton method
- Levenberg–Marquardt method

Nonlinear least squares

$$\text{minimize} \quad \sum_{i=1}^{m} f_i(x)^2 = \|f(x)\|^2$$

- $f_1(x), \ldots, f_m(x)$ are differentiable functions of a vector variable x
- f is a function from $\mathbf{R}^n$ to $\mathbf{R}^m$ with components $f_i(x)$:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

- problem reduces to (linear) least squares if $f(x) = Ax - b$

Location from range measurements

- vector x_{ex} represents unknown location in 2-D or 3-D
- we estimate x_{ex} by measuring distances to known points $a_1, \ldots, a_m$:

$$\rho_i = \|x_{\text{ex}} - a_i\| + v_i, \quad i = 1, \ldots, m$$

- v_i is measurement error

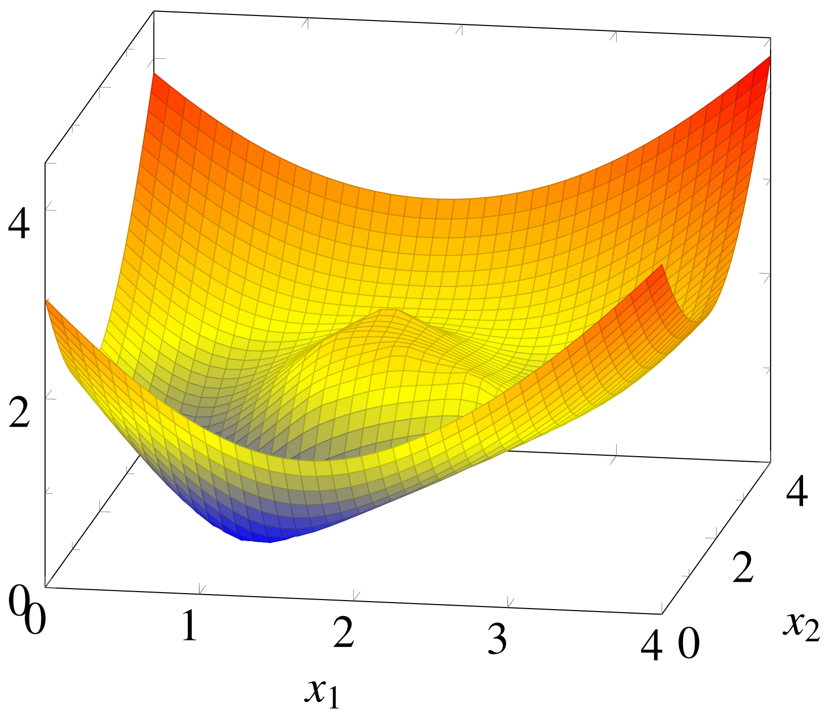
Nonlinear least squares estimate: compute estimate $\hat{x}$ by minimizing

$$\sum_{i=1}^{m} (\|x - a_i\| - \rho_i)^2$$

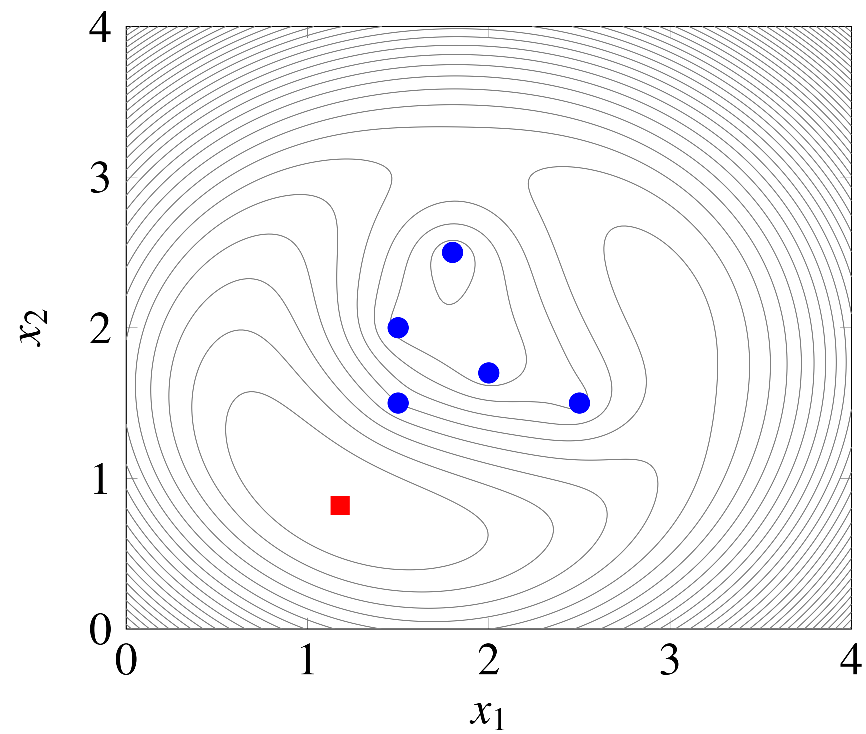
this is a nonlinear least squares problem with $f_i(x) = \|x - a_i\| - \rho_i$

Example

graph of $\|f(x)\|$

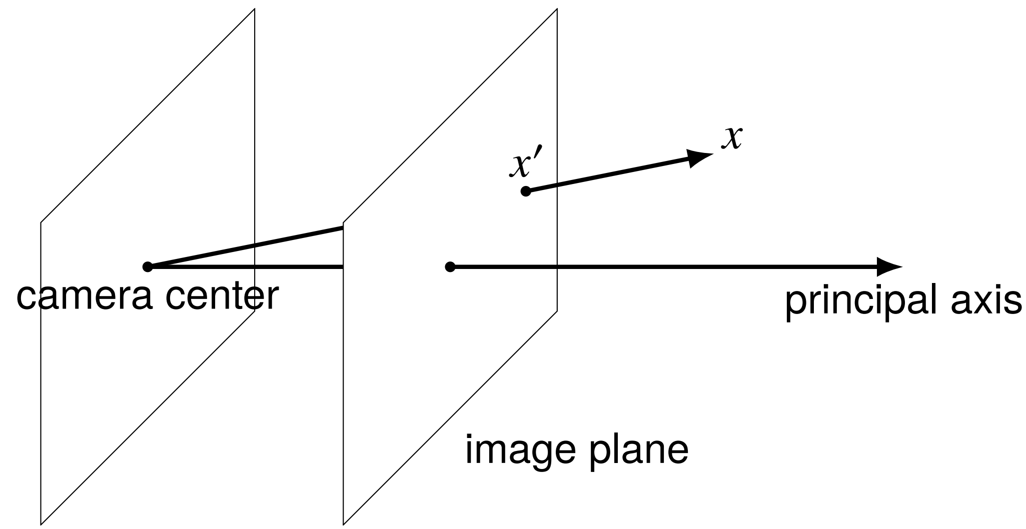


contour lines of $\|f(x)\|^2$



- correct position is $x_{\text{ex}} = (1, 1)$
- five points a_i, marked with blue dots
- red square marks nonlinear least squares estimate $\hat{x} = (1.18, 0.82)$

Location from multiple camera views



Camera model: described by parameters $A \in \mathbf{R}^{2\times3}$, $b \in \mathbf{R}^2$, $c \in \mathbf{R}^3$, $d \in \mathbf{R}$

- object at location $x \in \mathbf{R}^3$ creates image at location $x' \in \mathbf{R}^2$ in image plane

$$x' = \frac{1}{c^T x + d}(Ax + b)$$

  $c^T x + d > 0$ if object is in front of the camera

- A, b, c, d characterize the camera, and its position and orientation

Location from multiple camera views

- an object at location x_{ex} is viewed by l cameras (described by A_i, b_i, c_i, d_i)
- the image of the object in the image plane of camera i is at location

$$y_i = \frac{1}{c_i^T x_{\text{ex}} + d_i}(A_i x_{\text{ex}} + b_i) + v_i$$

- v_i is measurement or quantization error
- goal is to estimate 3-D location x_{ex} from the l observations $y_1, \ldots, y_l$

Nonlinear least squares estimate: compute estimate $\hat{x}$ by minimizing

$$\sum_{i=1}^{l} \left\| \frac{1}{c_i^T x + d_i}(A_i x + b_i) - y_i \right\|^2$$

this is a nonlinear least squares problem with $m = 2l$,

$$f_i(x) = \frac{(A_i x + b_i)_1}{c_i^T x + d_i} - (y_i)_1, \qquad f_{l+i}(x) = \frac{(A_i x + b_i)_2}{c_i^T x + d_i} - (y_i)_2$$

Model fitting

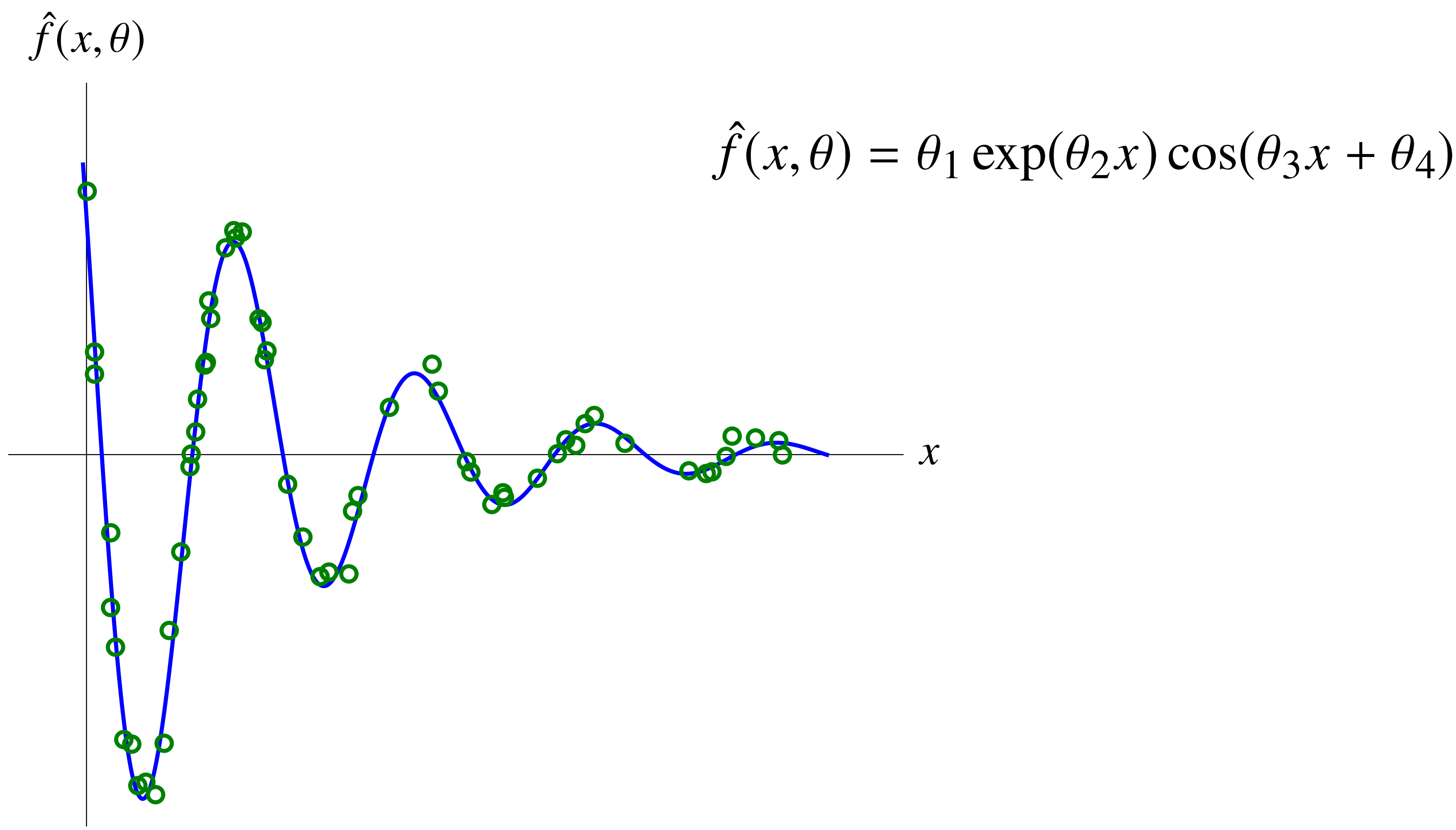
$$\text{minimize} \quad \sum_{i=1}^{N} (\hat{f}(x^{(i)}, \theta) - y^{(i)})^2$$

- model $\hat{f}(x, \theta)$ is parameterized by parameters $\theta_1, \ldots, \theta_p$
- $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$ are data points
- the minimization is over the model parameters θ
- on page 9.9 we considered models that are linear in the parameters θ:

$$\hat{f}(x, \theta) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

here we allow $\hat{f}(x, \theta)$ to be a nonlinear function of θ

Example



a nonlinear least squares problem with four variables θ_1, θ_2, θ_3, θ_4:

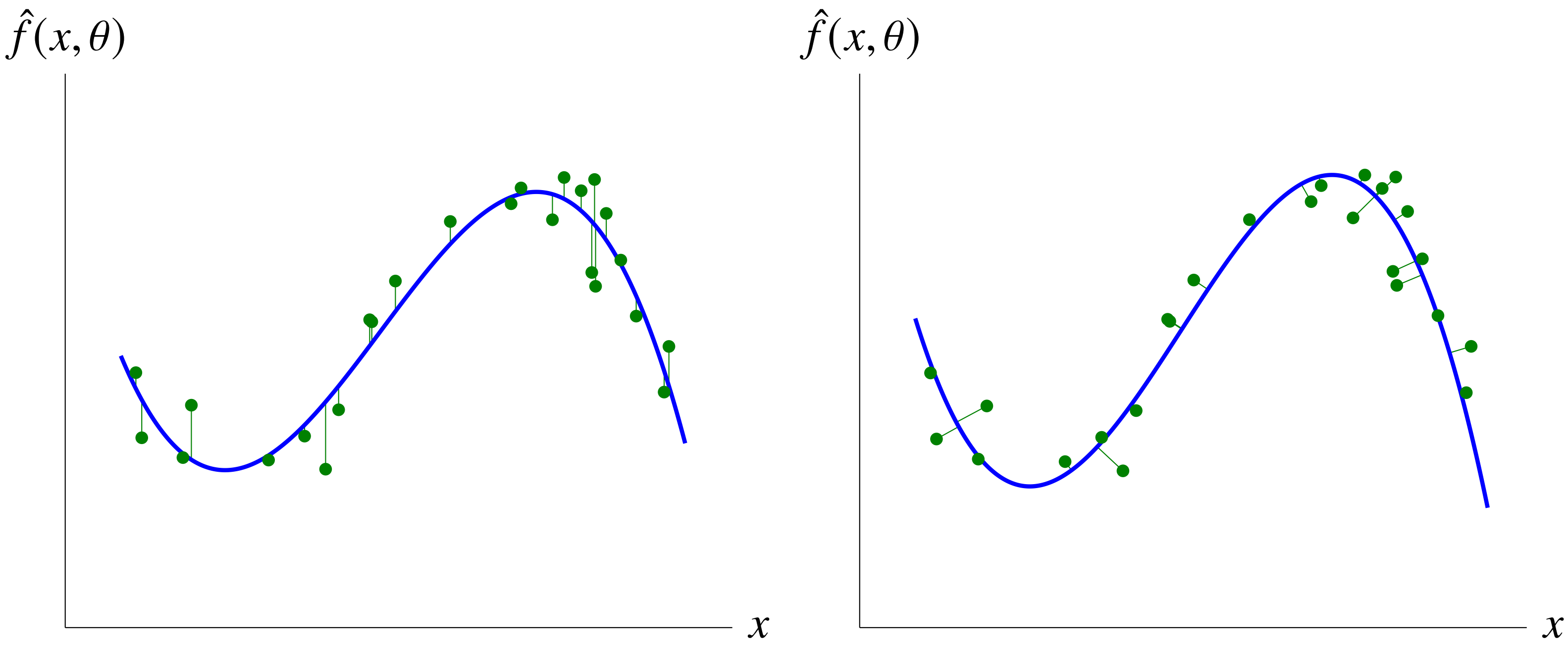
$$\text{minimize} \quad \sum_{i=1}^{N} \left(\theta_1 e^{\theta_2 x^{(i)}} \cos(\theta_3 x^{(i)} + \theta_4) - y^{(i)} \right)^2$$

Orthogonal distance regression

minimize the mean square distance of data points to graph of $\hat{f}(x, \theta)$

Example: orthogonal distance regression with cubic polynomial

$$\hat{f}(x, \theta) = \theta_1 + \theta_2 x + \theta_3 x^2 + \theta_4 x^3$$



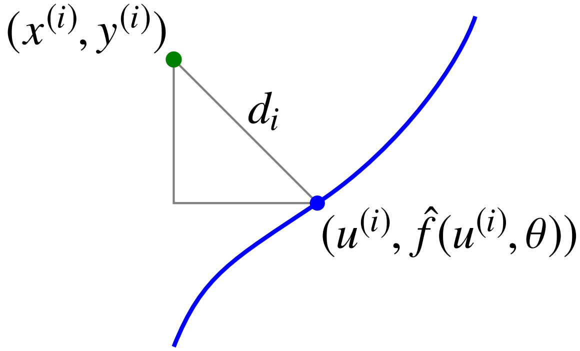
standard least squares fit

orthogonal distance fit

Nonlinear least squares formulation

$$\text{minimize} \quad \sum_{i=1}^{N} \left((\hat{f}(u^{(i)}, \theta) - y^{(i)})^2 + \|u^{(i)} - x^{(i)}\|^2 \right)$$

- optimization variables are model parameters θ and N points $u^{(i)}$
- ith term is squared distance of data point $(x^{(i)}, y^{(i)})$ to point $(u^{(i)}, \hat{f}(u^{(i)}, \theta))$



$$d_i^2 = (\hat{f}(u^{(i)}, \theta) - y^{(i)})^2 + \|u^{(i)} - x^{(i)}\|^2$$

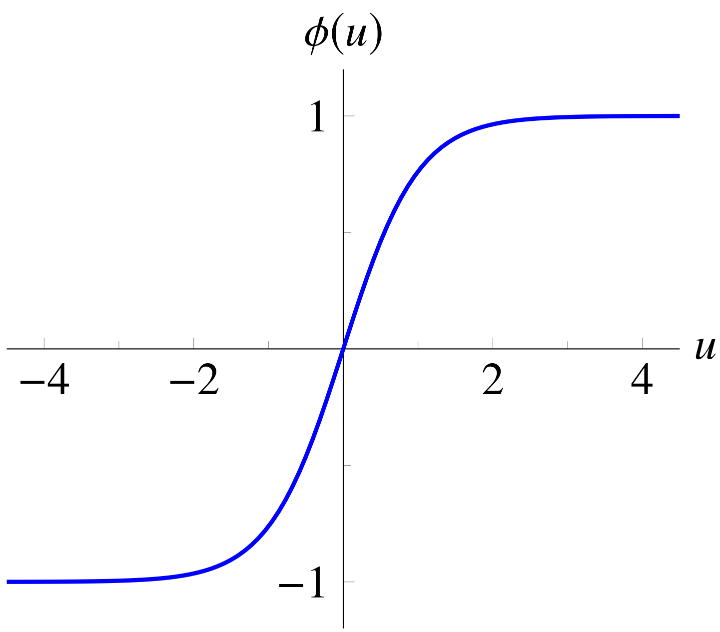
- minimizing d_i^2 over $u^{(i)}$ gives squared distance of $(x^{(i)}, y^{(i)})$ to graph
- minimizing $\sum_i d_i^2$ over $u^{(1)}, \ldots, u^{(N)}$ and θ minimizes mean squared distance

Binary classification

$$\hat{f}(x,\theta) = \mathrm{sign}\left(\theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p f_p(x)\right)$$

- in lecture 9 (p 9.25) we computed θ by solving a linear least squares problem
- better results are obtained by solving a nonlinear least squares problem

$$\text{minimize} \quad \sum_{i=1}^{N} \left(\phi(\theta_1 f_1(x^{(i)}) + \cdots + \theta_p f_p(x^{(i)})) - y^{(i)}\right)^2$$



- $(x^{(i)}, y^{(i)})$ are data points, $y^{(i)} \in \{-1, 1\}$
- $\phi(u)$ is the sigmoidal function

$$\phi(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

a differentiable approximation of $\mathrm{sign}(u)$

Outline

- definition and examples
- **derivatives and optimality condition**
- Gauss–Newton method
- Levenberg–Marquardt method

Gradient

Gradient of differentiable function $g : \mathbf{R}^n \to \mathbf{R}$ at $z \in \mathbf{R}^n$ is

$$\nabla g(z) = \left(\frac{\partial g}{\partial x_1}(z), \frac{\partial g}{\partial x_2}(z), \ldots, \frac{\partial g}{\partial x_n}(z) \right)$$

Affine approximation (linearization) of g around z is

$$\begin{aligned} \hat{g}(x) &= g(z) + \frac{\partial g}{\partial x_1}(z)(x_1 - z_1) + \cdots + \frac{\partial g}{\partial x_n}(z)(x_n - z_n) \\ &= g(z) + \nabla g(z)^T (x - z) \end{aligned}$$

(see page 1.27)

Derivative matrix

Derivative matrix (Jacobian) of differentiable function $f : \mathbf{R}^n \to \mathbf{R}^m$ at $z \in \mathbf{R}^n$:

$$Df(z) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(z) & \dfrac{\partial f_1}{\partial x_2}(z) & \cdots & \dfrac{\partial f_1}{\partial x_n}(z) \\ \dfrac{\partial f_2}{\partial x_1}(z) & \dfrac{\partial f_2}{\partial x_2}(z) & \cdots & \dfrac{\partial f_2}{\partial x_n}(z) \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial f_m}{\partial x_1}(z) & \dfrac{\partial f_m}{\partial x_2}(z) & \cdots & \dfrac{\partial f_m}{\partial x_n}(z) \end{bmatrix} = \begin{bmatrix} \nabla f_1(z)^T \\ \nabla f_2(z)^T \\ \vdots \\ \nabla f_m(z)^T \end{bmatrix}$$

Affine approximation (linearization) of f around z is

$$\hat{f}(x) = f(z) + Df(z)(x - z)$$

- see page 3.40
- we also use notation $\hat{f}(x; z)$ to indicate the point z around which we linearize

Gradient of nonlinear least squares cost

$$g(x) = \|f(x)\|^2 = \sum_{i=1}^{m} f_i(x)^2$$

- first derivative of g with respect to x_j:

$$\frac{\partial g}{\partial x_j}(z) = 2 \sum_{i=1}^{m} f_i(z) \frac{\partial f_i}{\partial x_j}(z)$$

- gradient of g at z:

$$\nabla g(z) = \begin{bmatrix} \dfrac{\partial g}{\partial x_1}(z) \\ \vdots \\ \dfrac{\partial g}{\partial x_n}(z) \end{bmatrix} = 2 \sum_{i=1}^{m} f_i(z) \nabla f_i(z) = 2Df(z)^T f(z)$$

Optimality condition

$$\text{minimize} \quad g(x) = \sum_{i=1}^{m} f_i(x)^2$$

- necessary condition for optimality: if x minimizes $g(x)$ then it must satisfy

$$\nabla g(x) = 2Df(x)^T f(x) = 0$$

- this generalizes the normal equations: if $f(x) = Ax - b$, then $Df(x) = A$ and

$$\nabla g(x) = 2A^T(Ax - b)$$

- for general f, the condition $\nabla g(x) = 0$ is not sufficient for optimality

Outline

- definition and examples
- derivatives and optimality condition
- **Gauss–Newton method**
- Levenberg–Marquardt method

Gauss–Newton method

$$\text{minimize} \quad g(x) = \|f(x)\|^2 = \sum_{i=1}^{m} f_i(x)^2$$

start at some initial guess $x^{(1)}$, and repeat for $k = 1, 2, \ldots$:

- linearize f around $x^{(k)}$:

$$\hat{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})$$

- substitute affine approximation $\hat{f}(x; x^{(k)})$ for f in least squares problem:

$$\text{minimize} \quad \|\hat{f}(x; x^{(k)})\|^2$$

- take the solution of this (linear) least squares problem as $x^{(k+1)}$

Gauss–Newton update

least squares problem solved in iteration k:

$$\text{minimize} \quad \|f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})\|^2$$

- if $Df(x^{(k)})$ has linearly independent columns, solution is given by

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)})\right)^{-1} Df(x^{(k)})^T f(x^{(k)})$$

- Gauss–Newton step $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ is

$$\begin{aligned}
\Delta x^{(k)} &= -\left(Df(x^{(k)})^T Df(x^{(k)})\right)^{-1} Df(x^{(k)})^T f(x^{(k)}) \\
&= -\frac{1}{2}\left(Df(x^{(k)})^T Df(x^{(k)})\right)^{-1} \nabla g(x^{(k)})
\end{aligned}$$

(using the expression for $\nabla g(x)$ on page 13.14)

Predicted cost reduction in iteration k

- predicted cost function at $x^{(k+1)}$, based on approximation $\hat{f}(x; x^{(k)})$:

$$
\begin{aligned}
&\|\hat{f}(x^{(k+1)}; x^{(k)})\|^2 \\
&\quad = \|f(x^{(k)}) + Df(x^{(k)})\Delta x^{(k)}\|^2 \\
&\quad = \|f(x^{(k)})\|^2 + 2f(x^{(k)})^T Df(x^{(k)})\Delta x^{(k)} + \|Df(x^{(k)})\Delta x^{(k)}\|^2 \\
&\quad = \|f(x^{(k)})\|^2 - \|Df(x^{(k)})\Delta x^{(k)}\|^2
\end{aligned}
$$

- if columns of $Df(x^{(k)})$ are linearly independent and $\Delta x^{(k)} \neq 0$,

$$\|\hat{f}(x^{(k+1)}; x^{(k)})\|^2 < \|f(x^{(k)})\|^2$$

- however, $\hat{f}(x; x^{(k)})$ is only a local approximation of $f(x)$, so it is possible that

$$\|f(x^{(k+1)})\|^2 > \|f(x^{(k)})\|^2$$

Outline

- definition and examples
- derivatives and optimality condition
- Gauss–Newton method
- **Levenberg–Marquardt method**

Levenberg–Marquardt method

addresses two difficulties in Gauss–Newton method:

- how to update $x^{(k)}$ when columns of $Df(x^{(k)})$ are linearly dependent
- what to do when the Gauss–Newton update does not reduce $\|f(x)\|^2$

Levenberg–Marquardt method

compute $x^{(k+1)}$ by solving a *regularized* least squares problem

$$\text{minimize} \quad \|\hat{f}(x; x^{(k)})\|^2 + \lambda^{(k)}\|x - x^{(k)}\|^2$$

- as before, $\hat{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})$
- second term forces x to be close to $x^{(k)}$ where $\hat{f}(x; x^{(k)}) \approx f(x)$
- with $\lambda^{(k)} > 0$, always has a unique solution (no condition on $Df(x^{(k)})$)

Levenberg–Marquardt update

regularized least squares problem solved in iteration k

$$\text{minimize} \quad \|f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})\|^2 + \lambda^{(k)}\|x - x^{(k)}\|^2$$

- solution is given by

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\right)^{-1} Df(x^{(k)})^T f(x^{(k)})$$

- Levenberg–Marquardt step $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ is

$$\begin{aligned}
\Delta x^{(k)} &= -\left(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\right)^{-1} Df(x^{(k)})^T f(x^{(k)}) \\
&= -\frac{1}{2}\left(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\right)^{-1} \nabla g(x^{(k)})
\end{aligned}$$

- for $\lambda^{(k)} = 0$ this is the Gauss–Newton step (if defined); for large $\lambda^{(k)}$,

$$\Delta x^{(k)} \approx -\frac{1}{2\lambda^{(k)}} \nabla g(x^{(k)})$$

Regularization parameter

several strategies for adapting $\lambda^{(k)}$ are possible; for example:

- at iteration k, compute the solution $\hat{x}$ of

$$\text{minimize} \quad \|\hat{f}(x; x^{(k)})\|^2 + \lambda^{(k)} \|x - x^{(k)}\|^2$$

- if $\|f(\hat{x})\|^2 < \|f(x^{(k)}\|^2$, take $x^{(k+1)} = \hat{x}$ and decrease λ
- otherwise, do not update x (take $x^{(k+1)} = x^{(k)}$), but increase λ

Some variations

- compare actual cost reduction with predicted cost reduction
- solve a least squares problem with "trust region"

$$\begin{array}{ll} \text{minimize} & \|\hat{f}(x; x^{(k)})\|^2 \\ \text{subject to} & \|x - x^{(k)}\|^2 \leq \gamma \end{array}$$

Summary: Levenberg–Marquardt method

choose $x^{(1)}$ and $\lambda^{(1)}$ and repeat for $k = 1, 2, \ldots$:

1. evaluate $f(x^{(k)})$ and $A = Df(x^{(k)})$
2. compute solution of regularized least squares problem:

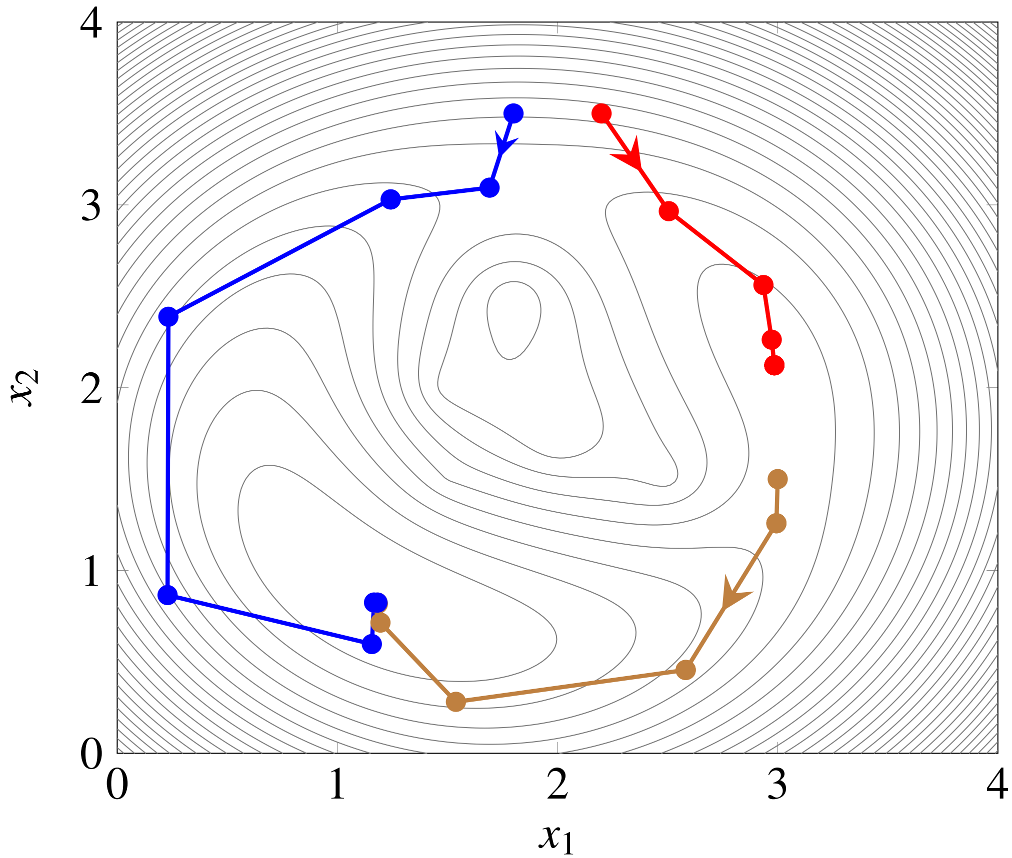
$$\hat{x} = x^{(k)} - (A^T A + \lambda^{(k)} I)^{-1} A^T f(x^{(k)})$$

3. define $x^{(k+1)}$ and $\lambda^{(k+1)}$ as follows:

$$\begin{cases} x^{(k+1)} = \hat{x} \text{ and } \lambda^{(k+1)} = \beta_1 \lambda^{(k)} & \text{if } \|f(\hat{x})\|^2 < \|f(x^{(k)})\|^2 \\ x^{(k+1)} = x^{(k)} \text{ and } \lambda^{(k+1)} = \beta_2 \lambda^{(k)} & \text{otherwise} \end{cases}$$

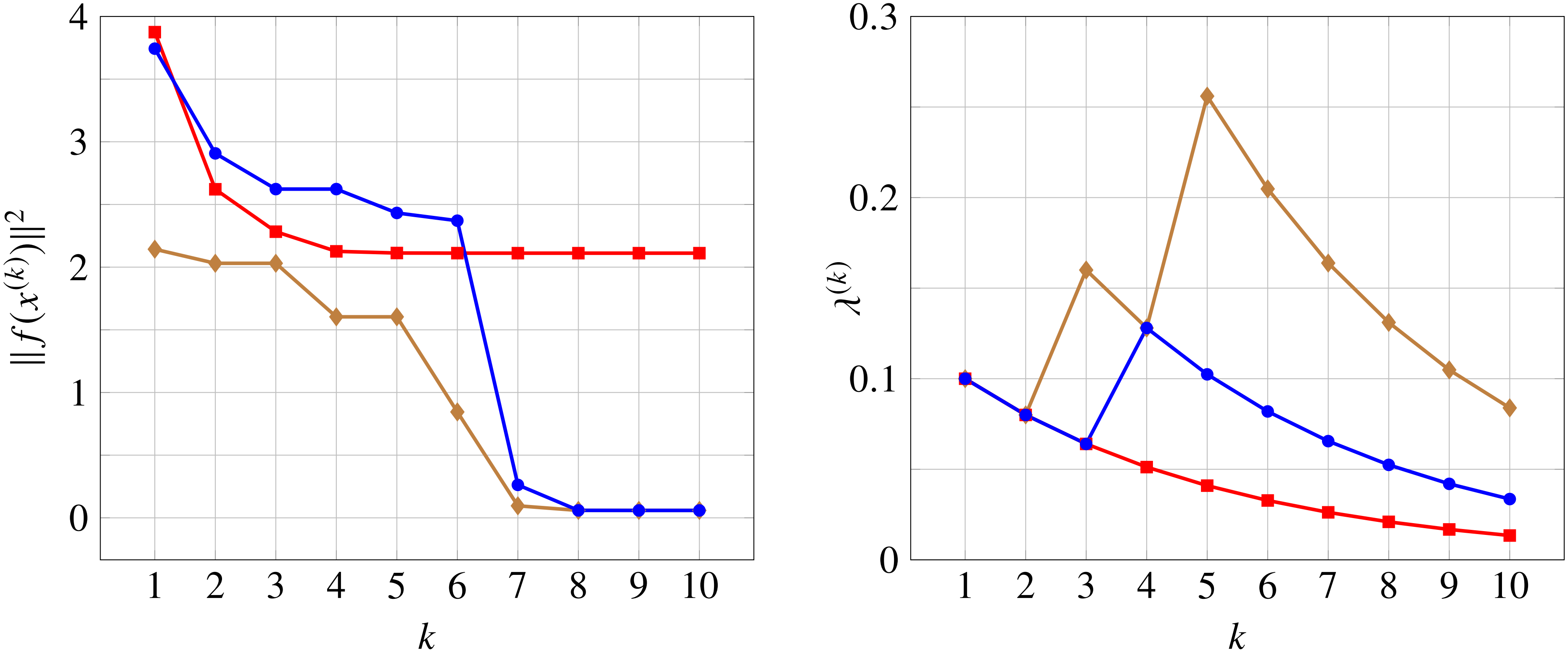
- β_1, β_2 are constants with $0 < \beta_1 < 1 < \beta_2$
- in step 2, $\hat{x}$ can be computed using a QR factorization
- terminate if $\nabla g(x^{(k)}) = 2A^T f(x^{(k)})$ is sufficiently small

Location from range measurements



- iterates from three starting points, with $\lambda^{(1)} = 0.1$, $\beta_1 = 0.8$, $\beta_2 = 2$
- algorithm started at $(1.8, 3.5)$ and $(3.0, 1.5)$ finds minimum $(1.18, 0.82)$
- started at $(2.2, 3.5)$ converges to non-optimal point

Cost function and regularization parameter



cost function and $\lambda^{(k)}$ for the three starting points on previous page