Advanced Statistical Computing
Lecture 2

Optimization & Solving Nonlinear Equation (II)

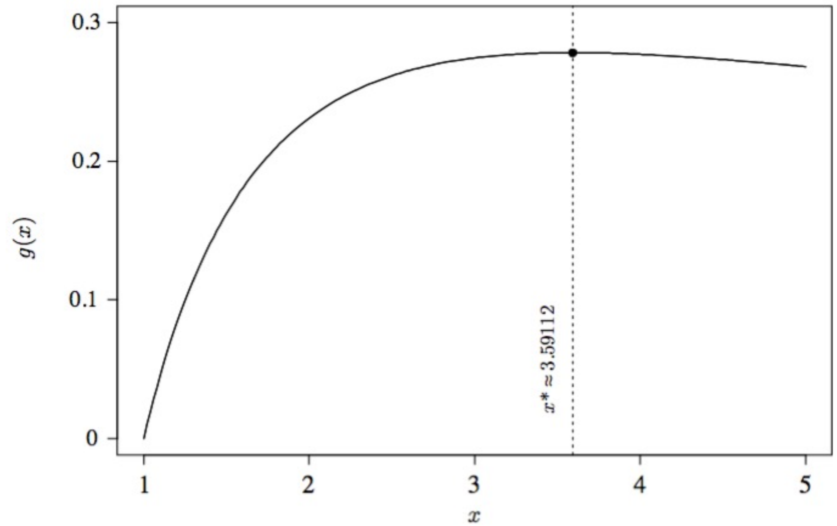
Dr. Ke Deng
Center for statistical Science
Tsinghua University, Beijing

邓 柯
清华大学统计学研究中心

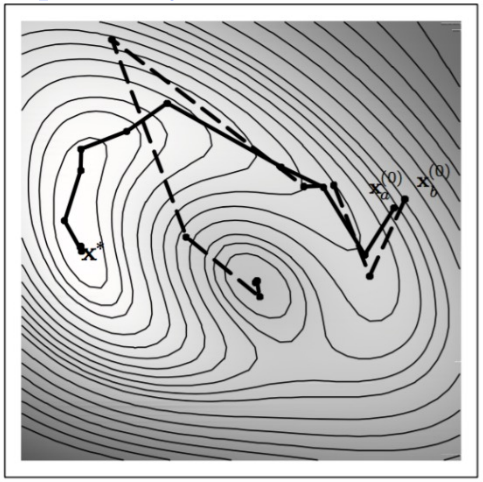
kdeng@tsinghua.edu.cn

From Univariate to Multivariate

Optimizing a univariate function



Optimizing a bivariate function



Multivariate optimization

- Seek the optimum of a real-valued function of a p-dimensional vector
- Many principles for the univariate case also apply for multivariate case
  - Algorithms are still iterative
  - Many algorithms take steps based on a local linearization of g' derived from a Taylor series or secant approximation
  - Convergence criteria are similar in spirit despite slight changes in form

Distance Measure & Convergence Criteria

- Distance measure for p-dimensional vectors: $D(\mathbf{u}, \mathbf{v})$
  - Two obvious choice:

$$D(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{p} |u_i - v_i|$$

$$D(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{p} (u_i - v_i)^2}$$

- Convergence Criteria:
  - Absolute convergence criteria:

$$D(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}) < \epsilon$$

  - Relative convergence criteria:

$$\frac{D(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)})}{D(\mathbf{x}^{(t)}, \mathbf{0})} < \epsilon \quad \text{or} \quad \frac{D(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)})}{D(\mathbf{x}^{(t)}, \mathbf{0}) + \epsilon} < \epsilon$$

M_1: Newton's Method (for univariate case)

❖ The motivating example:

Score equation with no analytic solution

Target function to maximize: $g(x) = \dfrac{\log x}{1+x} \Longrightarrow g'(x) = \dfrac{1 + 1/x - \log x}{(1+x)^2} = 0$

- The key idea:
  - Approximated nonlinear score equation with a linear equation nearby $x^{(t)}$:

$$0 = g'(x^*) \approx g'(x^{(t)}) + (x^* - x^{(t)})g''(x^{(t)})$$

  - Get solution: $x^* = x^{(t)} - \dfrac{g'(x^{(t)})}{g''(x^{(t)})}$

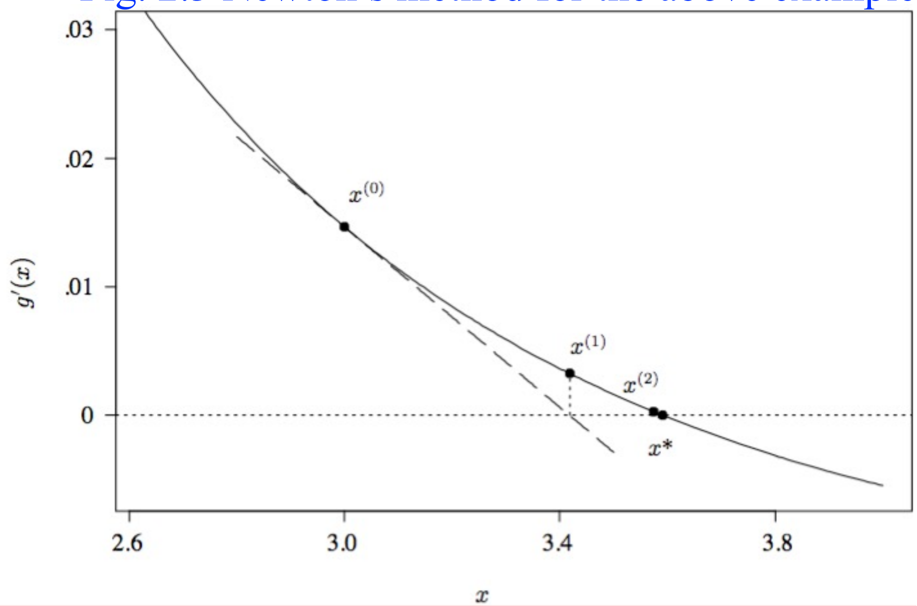
- Updating equation:

$$-\frac{g'(x^{(t)})}{g''(x^{(t)})}$$

$$x^{(t+1)} = x^{(t)} - \frac{g'(x^{(t)})}{g''(x^{(t)})} = x^{(t)} + h^{(t)}$$

- Technical conditions:
  - g' is continuously differentiable
  - $g''(x^*) \neq 0$

Fig. 2.3 Newton's method for the above example



$$h^{(t)} = \frac{(x^{(t)}+1)(1 + 1/x^{(t)} - \log x^{(t)})}{3 + 4/x^{(t)} + 1/(x^{(t)})^2 - 2\log x^{(t)}}$$ for the motivating example

M_1: Newton's Method (for univariate case)

❖ The motivating example:

Score equation with no analytic solution

Target function to maximize: $g(x) = \frac{\log x}{1+x} \Longrightarrow g'(x) = \frac{1 + 1/x - \log x}{(1+x)^2} = 0$

- The key idea:

  ➢ Approximated nonlinear score equation with a linear equation nearby $x^{(t)}$:

  $$0 = g'(x^*) \approx g'(x^{(t)}) + (x^* - x^{(t)})g''(x^{(t)})$$

  ➢ Get solution: $x^* = x^{(t)} - \frac{g'(x^{(t)})}{g''(x^{(t)})}$

An alternative perspective

➢ Approximated target function g with a quadratic Taylor expansion nearby $x^{(t)}$:

$$g(x^{(t)}) + (x^* - x^{(t)})g'(x^{(t)}) + (x^* - x^{(t)})^2 g''(x^{(t)})/2$$

➢ Maximize the quadratic function:

$$x^* = x^{(t)} - \frac{g'(x^{(t)})}{g''(x^{(t)})}$$

- Updating equation:

$$x^{(t+1)} = x^{(t)} - \frac{g'(x^{(t)})}{g''(x^{(t)})} = x^{(t)} + \underbrace{h^{(t)}}_{-\frac{g'(x^{(t)})}{g''(x^{(t)})}}$$

$$h^{(t)} = \frac{(x^{(t)}+1)(1 + 1/x^{(t)} - \log x^{(t)})}{3 + 4/x^{(t)} + 1/(x^{(t)})^2 - 2\log x^{(t)}}$$ for the motivating example

- Technical conditions:

  ➢ g' is continuously differentiable

  ➢ $g''(x^*) \neq 0$

M_1: Newton's Method (for multivariate case)

- Similar idea as in the univariate case:
  - Approximate target function g with a quadratic Taylor expansion nearby $x^{(t)}$:

$$g(\mathbf{x}^*) \approx g(\mathbf{x}^{(t)}) + (\mathbf{x}^* - \mathbf{x}^{(t)})^{\mathrm{T}} \mathbf{g}'(\mathbf{x}^{(t)}) + \frac{1}{2}(\mathbf{x}^* - \mathbf{x}^{(t)})^{\mathrm{T}} \mathbf{g}''(\mathbf{x}^{(t)})(\mathbf{x}^* - \mathbf{x}^{(t)})$$

  - Maximize the above quadratic function with respect to $\boldsymbol{x}^*$ to find the next iterate

$$\Updownarrow$$

  - Set the gradient of the right-hand side to zero: $\mathbf{g}'(\mathbf{x}^{(t)}) + \mathbf{g}''(\mathbf{x}^{(t)})(\mathbf{x}^* - \mathbf{x}^{(t)}) = 0$

$$\Downarrow$$

  - Get updating function:

$$\mathbf{h}^{(t)} = -\mathbf{g}''(\mathbf{x}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{g}''(\mathbf{x}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)}) = \mathbf{x}^{(t)} + \mathbf{h}^{(t)}$$

- M_1': Fisher Scoring for finding MLE in multivariate case:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \mathbf{I}(\boldsymbol{\theta}^{(t)})^{-1}\mathbf{l}'(\boldsymbol{\theta}^{(t)})$$

Replace the observed information $-g''(\theta^{(t)})$ with the expected Fisher information $\boldsymbol{I}(\theta^{(t)})$

Bivariate Optimization via M_1
An Example

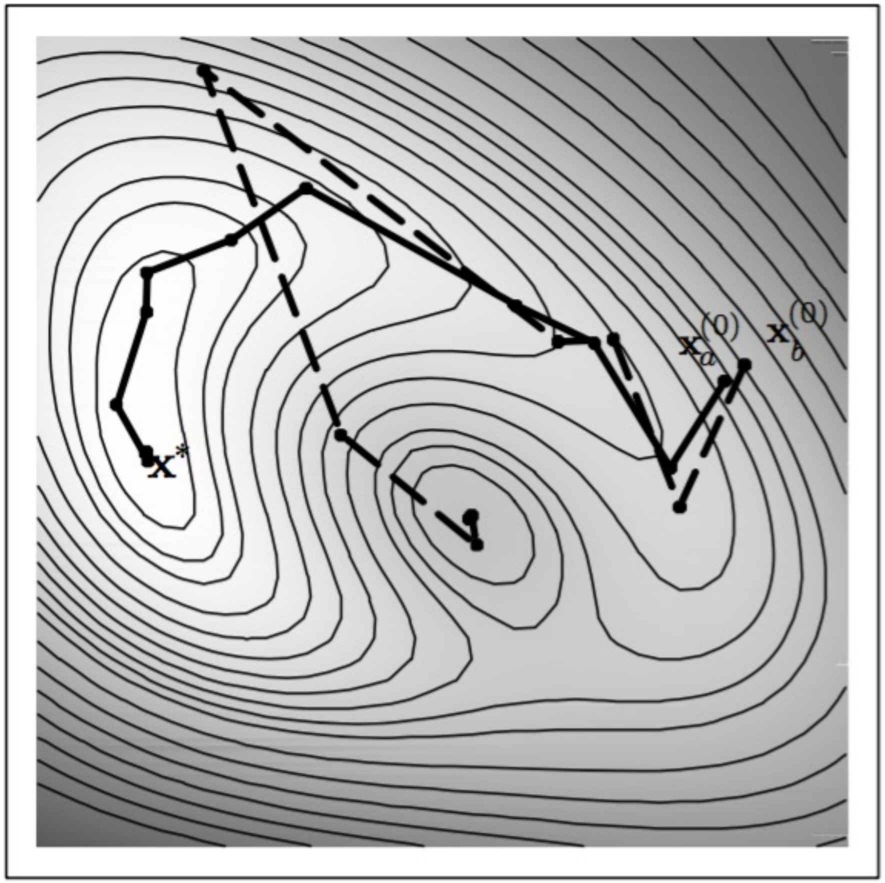


Fig. 2.7 Newton's method with different starting points

Observations

- Two starting points are tried:
  - ✓ one converges to the true maximum
  - ✓ one converges to a local maximum
- Some steps are downhill
- Some steps have a large step length

Impression

- Even a small change of starting point may lead to very different results
- Large step length may overshoot the portion of the ridge

M_1 and M_1' in Logistic Regression

- Key assumptions of Logistic Regression:

$$Y_i|\mathbf{z}_i \sim \text{Bernoulli}(\pi_i) \text{ independently for } i = 1, \ldots, n$$

$$\theta_i = \log\{\pi_i/(1-\pi_i)\}$$

$$\theta_i = \mathbf{z}_i^{\mathrm{T}}\boldsymbol{\beta} \qquad \mathbf{z}_i = (1,\ z_i)^{\mathrm{T}} \qquad \boldsymbol{\beta} = (\beta_0,\ \beta_1)^{\mathrm{T}}$$

- Log-likelihood:

$n \times 2$ matrix whose ith row is $\mathbf{z}_i^{\mathrm{T}}$

$$l(\boldsymbol{\beta}) = \mathbf{y}^{\mathrm{T}}\mathbf{Z}\boldsymbol{\beta} - \mathbf{b}^{\mathrm{T}}\mathbf{1}$$

$\mathbf{y} = (y_1 \ldots y_n)^{\mathrm{T}}$ $\qquad$ $\mathbf{b} = (b(\theta_1) \ldots b(\theta_n))^{\mathrm{T}}$

$$b(\theta_i) = \log\{1 + \exp\{\theta_i\}\} = \log\{1 + \exp\{\mathbf{z}_i^{\mathrm{T}}\boldsymbol{\beta}\}\} = -\log\{1 - \pi_i\}$$

M_1 and M_1' in Logistic Regression

- Newton's method:

$$\mathbf{l}'(\boldsymbol{\beta}) = \mathbf{Z}^{\mathrm{T}}(\mathbf{y} - \boldsymbol{\pi})$$

$\mathbf{W}$ is a diagonal matrix with ith diagonal entry equal to $\pi_i(1-\pi_i)$

$$\mathbf{l}''(\boldsymbol{\beta}) = \frac{d}{d\boldsymbol{\beta}}(\mathbf{Z}^{\mathrm{T}}(\mathbf{y} - \boldsymbol{\pi})) = -\left(\frac{d\boldsymbol{\pi}}{d\boldsymbol{\beta}}\right)^{\mathrm{T}} \mathbf{Z} = -\mathbf{Z}^{\mathrm{T}}\mathbf{W}\mathbf{Z}$$

does not depend on $\mathbf{y}$

$$\begin{aligned}\boldsymbol{\beta}^{(t+1)} &= \boldsymbol{\beta}^{(t)} - \mathbf{l}''(\boldsymbol{\beta}^{(t)})^{-1}\mathbf{l}'(\boldsymbol{\beta}^{(t)}) \\ &= \boldsymbol{\beta}^{(t)} + \left(\mathbf{Z}^{\mathrm{T}}\mathbf{W}^{(t)}\mathbf{Z}\right)^{-1}\left(\mathbf{Z}^{\mathrm{T}}(\mathbf{y} - \boldsymbol{\pi}^{(t)})\right)\end{aligned}$$

- Notes
  - The Hessian does not depend on $\mathbf{y}$ in this example
  - Therefore, Fisher's information matrix is equal to the observed information:

$$\boldsymbol{I}(\beta) = E\{-\boldsymbol{l}''(\beta)\} = E\{\boldsymbol{Z}^T\boldsymbol{W}\boldsymbol{Z}\} = -\boldsymbol{l}''(\beta).$$

  - Fisher scoring approach M_2' = Newton's method M_2 in this example.
  - $M_1 = M_1' =$ *iteratively reweighted least squares* (IRLS) $\subset$ Gauss-Newton method M_2

Example: Human Face Recognition

- Problem:

logistic regression model to some data related to testing a human face recognition algorithm

- Original data:

Pairs of images of 1072 faces

- Response:

$y_i = 1$: a successful match
$y_i = 0$: a match to another person

- Predictor variable:

absolute difference in mean standardized eye region pixel intensity between the probe image and its corresponding target

TABLE 2.1 Parameter estimates and corresponding variance– covariance matrix estimates are shown for each Newton's method iteration for fitting a logistic regression model to the face recognition data described in Example 2.5.

Iteration, t	$\boldsymbol{\beta}^{(t)}$	$-\mathbf{l}''(\boldsymbol{\beta}^{(t)})^{-1}$
0	$\begin{pmatrix} 0.95913 \\ 0.00000 \end{pmatrix}$	$\begin{pmatrix} 0.01067 & -0.11412 \\ -0.11412 & 2.16701 \end{pmatrix}$
1	$\begin{pmatrix} 1.70694 \\ -14.20059 \end{pmatrix}$	$\begin{pmatrix} 0.13312 & -0.14010 \\ -0.14010 & 2.36367 \end{pmatrix}$
2	$\begin{pmatrix} 1.73725 \\ -13.56988 \end{pmatrix}$	$\begin{pmatrix} 0.01347 & -0.13941 \\ -0.13941 & 2.32090 \end{pmatrix}$
3	$\begin{pmatrix} 1.73874 \\ -13.58839 \end{pmatrix}$	$\begin{pmatrix} 0.01349 & -0.13952 \\ -0.13952 & 2.32241 \end{pmatrix}$
4	$\begin{pmatrix} 1.73874 \\ -13.58840 \end{pmatrix}$	$\begin{pmatrix} 0.01349 & -0.13952 \\ -0.13952 & 2.32241 \end{pmatrix}$

Newton-Like Methods

- Newton method:

Hessian matrix

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{g}''(\mathbf{x}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

- Newton-like method:

a $p \times p$ matrix approximating the Hessian $\boldsymbol{g}''(\boldsymbol{x}^{(t)})$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - (\mathbf{M}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

- Reasons to replace the Hessian by some simpler approximation:
  - Computationally cheaper
  - A suitable $\boldsymbol{M}^{(t)}$ can guarantee ascent, which may be violated in Newton's method, i.e., $g(\boldsymbol{x}^{(t+1)}) > g(\boldsymbol{x}^{(t)})$

- Examples of Newton-like methods:
  - M_1' : Fisher scoring
  - M_1'' : Ascent algorithms
  - M_1''': Discrete Newton and Fixed-point methods
  - M_1'''':Quasi-Newton methods

M_1'': Ascent Algorithms

- Newton-like method:

a $p \times p$ matrix approximating the Hessian $\boldsymbol{g}''(\boldsymbol{x}^{(t)})$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - (\mathbf{M}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

- Key idea of M_1'':

  design a Newton-like method that guarantees uphill in every step

- Updating function of M_1'':

a *negative definite* matrix to approximate the Hessian

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \mathbf{h}^{(t)} \longrightarrow \mathbf{h}^{(t)} = -\alpha^{(t)}\left[\mathbf{M}^{(t)}\right]^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

a *contraction* or *step length* parameter $\alpha^{(t)} > 0$ whose value can shrink to ensure ascent at each step

- *Steepest Ascent* as a special case:

$$\mathbf{M}^{(t)} = -\mathbf{I} \Longrightarrow \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)})$$

Why M_1'' Works?

- Analysis of the error:

$$g(\mathbf{x}^{(t+1)}) - g(\mathbf{x}^{(t)}) = g(\mathbf{x}^{(t)} + \mathbf{h}^{(t)}) - g(\mathbf{x}^{(t)})$$

negligible when $\alpha^{(t)}$ is close to zero

Linear Taylor expansion of g $\rightarrow$ $= -\alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)})^{\mathrm{T}}(\mathbf{M}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)}) + \mathcal{O}(\alpha^{(t)})$

> 0 if $\boldsymbol{M}^{(t)}$ is negative define

- Theoretical properties of M_1'':

  A small enough step length will guarantee uphill in the framework of M_2''

- Tune $\alpha^{(t)}$ by *Step Halving*:
  - start each step with $\alpha^{(t)} = 1$.
  - If the original step turns out to be downhill, $\alpha^{(t)}$ can be halved.
  - If the step is still downhill, $\alpha^{(t)}$ is halved again until a sufficiently small step is found to be uphill.

Alternative Ways to Tune Step Length

- Step halving:

  Halve the step length at each time

- Line search methods:

  Search along the step direction (by any method) to find a proper step length

Backtracking

Practical Notes

- Backtracking with a positive definite replacement for the negative Hessian is not sufficient to ensure convergence of the algorithm, even when g is bounded above with a unique maximum
- It is also necessary to ensure the following two conditions:
  - steps make a sufficient ascent, i.e.,

    $g(\boldsymbol{x}^{(t)}) - g(\boldsymbol{x}^{(t-1)})$ does not decrease too quickly as t increases
  - step directions are not nearly orthogonal to the gradient, i.e., avoid following a level contour of g

M_1''': Discrete Newton & Fixed-Point Methods

- Newton-like method:

a $p \times p$ matrix approximating the Hessian $\boldsymbol{g}''(\boldsymbol{x}^{(t)})$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \left(\mathbf{M}^{(t)}\right)^{-1} \mathbf{g}'(\mathbf{x}^{(t)})$$

- Multivariate discrete Newton method

$$\mathbf{M}_{ij}^{(t)} = \frac{g_i'\left(\mathbf{x}^{(t)} + h_{ij}^{(t)} \mathbf{e}_j\right) - g_i'\left(\mathbf{x}^{(t)}\right)}{h_{ij}^{(t)}}$$

  - avoid calculating the Hessian precisely by resorting to a secant-like method
  - computational burden is still heavy: at each step, $\boldsymbol{M}^{(t)}$ is wholly updated by calculating a new discrete difference for each element

- Multivariate fixed-point method

$$\mathbf{M}^{(t)} = \mathbf{M}$$

  - avoid calculating the Hessian by relying on an initial approximation
  - a reasonable choice for $\mathbf{M}$ is $\boldsymbol{g}''(\boldsymbol{x}^{(0)})$
  - computationally cheap

- If $\mathbf{M}$ is diagonal, equivalent to running univariate scaled fixed-point algorithm separately to each component of $\boldsymbol{g}$

M_1'''': Quasi-Newton Methods

- M_1''': Multivariate discrete Newton method

a $p \times p$ matrix approximating the Hessian $\boldsymbol{g}''(\boldsymbol{x}^{(t)})$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - (\mathbf{M}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

$$\mathbf{M}_{ij}^{(t)} = \frac{g_i'\left(\mathbf{x}^{(t)} + h_{ij}^{(t)}\mathbf{e}_j\right) - g_i'\left(\mathbf{x}^{(t)}\right)}{h_{ij}^{(t)}}$$

- avoid calculating the Hessian precisely by resorting to a secant-like method
- computational burden is still heavy

- The key idea of Quasi-Newton methods M_1'''':
  - abandon the component-wise discrete-difference approximation to $\mathbf{g}''$
  - approximate $\mathbf{g}''$ by secant condition based on differences

$$\mathbf{g}'(\mathbf{x}^{(t+1)}) - \mathbf{g}'(\mathbf{x}^{(t)}) = \mathbf{M}^{(t+1)}(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})$$

  - A equation about $\boldsymbol{M}^{(t+1)}$: # of constraints$= p$, # of parameters$= p^2$
  - Has unique solution when $p = 1$, i.e., the Ascent Method

Approximate $\boldsymbol{M}^{(t+1)}$ based on $\boldsymbol{M}^{(t)}$

- Unique symmetric rank-one method

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} + c^{(t)}\mathbf{v}^{(t)}(\mathbf{v}^{(t)})^{\mathrm{T}}$$

$$\mathbf{z}^{(t)} = \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)} \qquad \mathbf{y}^{(t)} = \mathbf{g}'(\mathbf{x}^{(t+1)}) - \mathbf{g}'(\mathbf{x}^{(t)})$$

$$\mathbf{v}^{(t)} = \mathbf{y}^{(t)} - \mathbf{M}^{(t)}\mathbf{z}^{(t)} \qquad c^{(t)} = 1/[(\mathbf{v}^{(t)})^{\mathrm{T}}\mathbf{z}^{(t)}]$$

- Broyden class rank-two methods

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} - \frac{\mathbf{M}^{(t)}\mathbf{z}^{(t)}(\mathbf{M}^{(t)}\mathbf{z}^{(t)})^{\mathrm{T}}}{(\mathbf{z}^{(t)})^{\mathrm{T}}\mathbf{M}^{(t)}\mathbf{z}^{(t)}} + \frac{\mathbf{y}^{(t)}(\mathbf{y}^{(t)})^{\mathrm{T}}}{(\mathbf{z}^{(t)})^{\mathrm{T}}\mathbf{y}^{(t)}} + \delta^{(t)}\left((\mathbf{z}^{(t)})^{\mathrm{T}}\mathbf{M}^{(t)}\mathbf{z}^{(t)}\right)\mathbf{d}^{(t)}(\mathbf{d}^{(t)})^{\mathrm{T}}$$

BFGS update: $\delta^{(t)} = 0$

$\delta^{(t)} = 1$

$$\mathbf{d}^{(t)} = \frac{\mathbf{y}^{(t)}}{(\mathbf{z}^{(t)})^{\mathrm{T}}\mathbf{y}^{(t)}} - \frac{\mathbf{M}^{(t)}\mathbf{z}^{(t)}}{(\mathbf{z}^{(t)})^{\mathrm{T}}\mathbf{M}^{(t)}\mathbf{z}^{(t)}}$$

Note:

- The order of convergence of quasi-Newton is usually between linear and quadratic.
- It’s fast and powerful, and widely used in popular software packages.
- It rely on the notion that the root-finding problem can be solved efficiently even using poor approximations to the Hessian.

Steepest Ascent vs Quasi-Newton

: Steepest Ascent

: Quasi-Newton with BFGS update

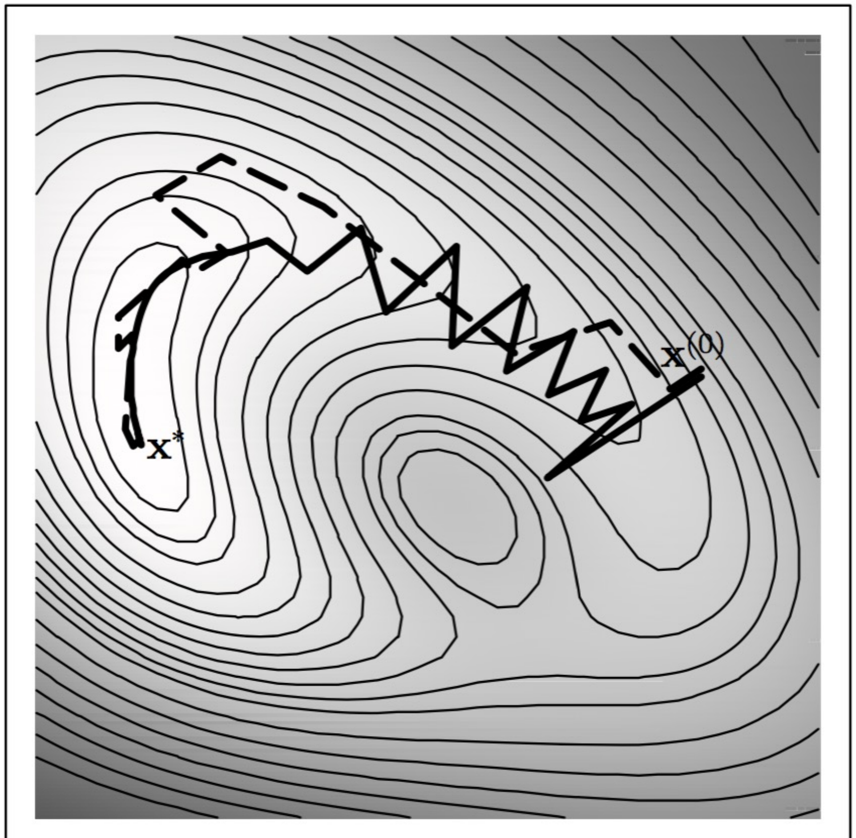


Fig. 2.8 Steepest Ascent & Quasi-Newton with the same starting point
(backtracking is employed by both methods)

Observations

- Both methods find the optimum
- Both methods guarantee uphill
- Steepest Ascent takes more steps
- Quasi-Newton seems have a better global view

Enhance the Performance & Stability of Quasi-Newton

- Choose a good starting matrix $\boldsymbol{M}^{(0)}$
  - The easiest choice is the negative identity matrix, i.e., $\boldsymbol{M}^{(0)} = -\boldsymbol{I}$ (but this is often inadequate if the scales of the components of $\mathbf{x}$(t) differ greatly)
  - In MLE problems, setting $\boldsymbol{M}^{(0)} = -\boldsymbol{I}(\theta^{(0)})$ is a much better choice

- Rescale the problem for a better performance
  - Rescale the problem so that the elements of $\mathbf{x}$ are on comparable scales
  - Fail to do so may lead in bad or unpredictable performance of the method

Improve the Estimation of Hessian

- Motivation
  - In the context of MLE and statistical inference, the Hessian is critical because it provides estimates of standard error and covariance
- Why improvement is needed?
  - The raw approximation of Hessian from Quasi-Newton may be quite bad
  - Quasi-Newton does not rely on a precise approximation of Hessian
  - If stopped at iteration t, the most recent Hessian approximation $\boldsymbol{M}^{(t-1)}$ is out of date and mislocated at $\theta^{(t-1)}$ instead of at $\theta^{(t)}$
- Key idea
  - compute a more precise approximation after iterations have stopped.
- An approach based on the central difference approximation:

$$\widehat{\mathbf{l}''\left(\boldsymbol{\theta}^{(t)}\right)} = \frac{l_i'(\boldsymbol{\theta}^{(t)} + h_{ij}\mathbf{e}_j) - l_i'(\boldsymbol{\theta}^{(t)} - h_{ij}\mathbf{e}_j)}{2h_{ij}}$$

Discrete approximation with respect to the latest guess of optimum $\theta^{(t)}$

M_2: Gauss-Newton Method

- Problem setting: optimizing a special case of functions of the form

$$g(\boldsymbol{\theta}) = -\sum_{i=1}^{n} (y_i - f(\mathbf{z}_i, \boldsymbol{\theta}))^2$$

A known (nonlinear) function

An important problem in regression

$$Y_i = f(\mathbf{z}_i, \boldsymbol{\theta}) + \epsilon_i$$

- Key idea of M_2 :
  - ➢ approximate f instead of g linearly by Taylor expansion at $\boldsymbol{\theta}^{(t)}$

$$Y_i \approx f(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^{\mathrm{T}} \mathbf{f}'(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}) + \epsilon_i = \tilde{f}(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}) + \epsilon_i$$

  - ➢ maximize the surrogate target function below with respect to $\boldsymbol{\theta}$

$$\Updownarrow \quad \tilde{g}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y_i - \tilde{f}(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}) \right]^2$$

  - ➢ find LSE $\boldsymbol{\theta}^*$ in a linear regression with $\boldsymbol{\theta}$ as the parameter
  - ➢ Updating function: $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^*$

- Practical notes for M_2:
  - ➢ It does not require computation of the Hessian
  - ➢ It is fast when f is nearly linear or when the model fits well
  - ➢ It may converge very slowly or not at all if the model fits poorly

M_3: Nonlinear Gauss-Seidel Iteration

- Key idea of M_3 :
  - Optimize one dimension at each time
  - Similar to Gibbs sampling
- Also known as:
  - Backfitting, or
  - Cyclic coordinate ascent
- Advantages:
  - Applying univariate optimization repeatedly
  - Easy to automate & program

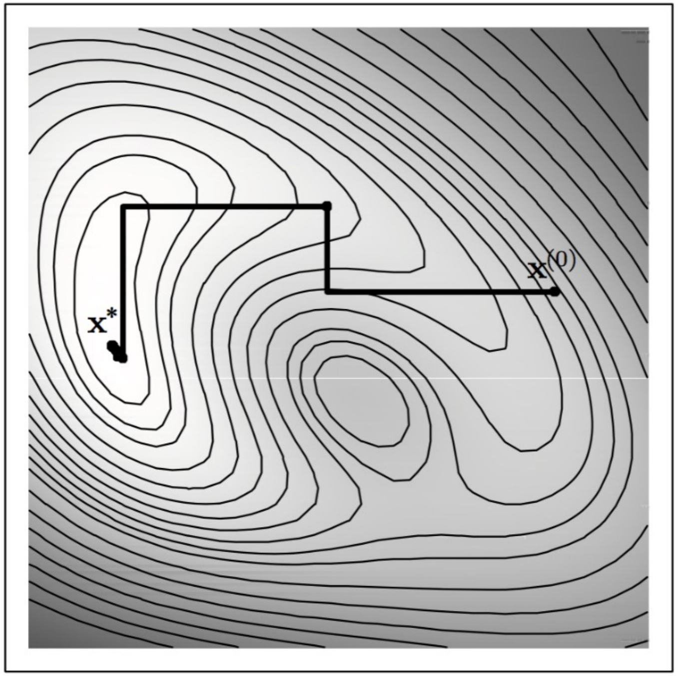


Fig. 2.14 The first a few steps of M_3

M_4: Bisection Method (for univariate case)

Suppose:

- g' is continuous on $[a_0, b_0]$
- $g'(a_0)g'(b_0) \leq 0$

$\xrightarrow[\text{theorem}]{\text{intermediate value}}$

- there exists at least one x^* in $[a_0, b_0]$ for which $g'(x^*) = 0$
- hence x^* is a local optimum of g

Bisection method constructs a sequence of nested intervals to capture x^*:

$$[a_0, b_0] \supset [a_1, b_1] \supset [a_2, b_2] \supset \cdots \text{ and so forth}$$

Note: direct application of this method becomes inefficient in multivariate case.

Starting value: $x^{(0)} = (a_0 + b_0)/2$

Updating equations:

$$[a_{t+1}, b_{t+1}] = \begin{cases} [a_t, x^{(t)}] & \text{if } g'(a_t)g'(x^{(t)}) \leq 0, \\ [x^{(t)}, b_t] & \text{if } g'(a_t)g'(x^{(t)}) > 0 \end{cases}$$

$$x^{(t)} = \frac{1}{2}(a_t + b_t) \Longleftrightarrow x^{(t)} = a_t + (b_t - a_t)/2$$

numerically more stable

Note: if g has more than one root in the starting interval, it is easy to see that bisection will find one of them, but will not find the rest.

M_4': Nelder–Mead Algorithm

- An iterative direct search approach
- Tries to nominate a superior point for the next iteration based on a collection of function evaluations at possible solutions while
- A smart version of the Bracketing Method for univariate case

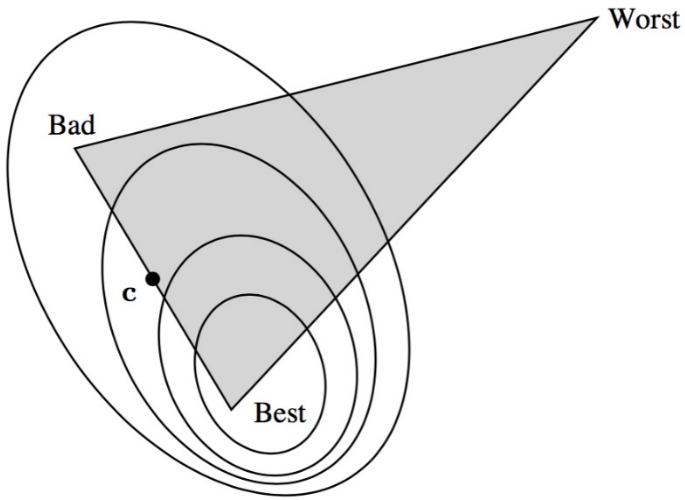


Fig. 2.9 Search for the optimum with a sequence of simplex

Five Types of Moves

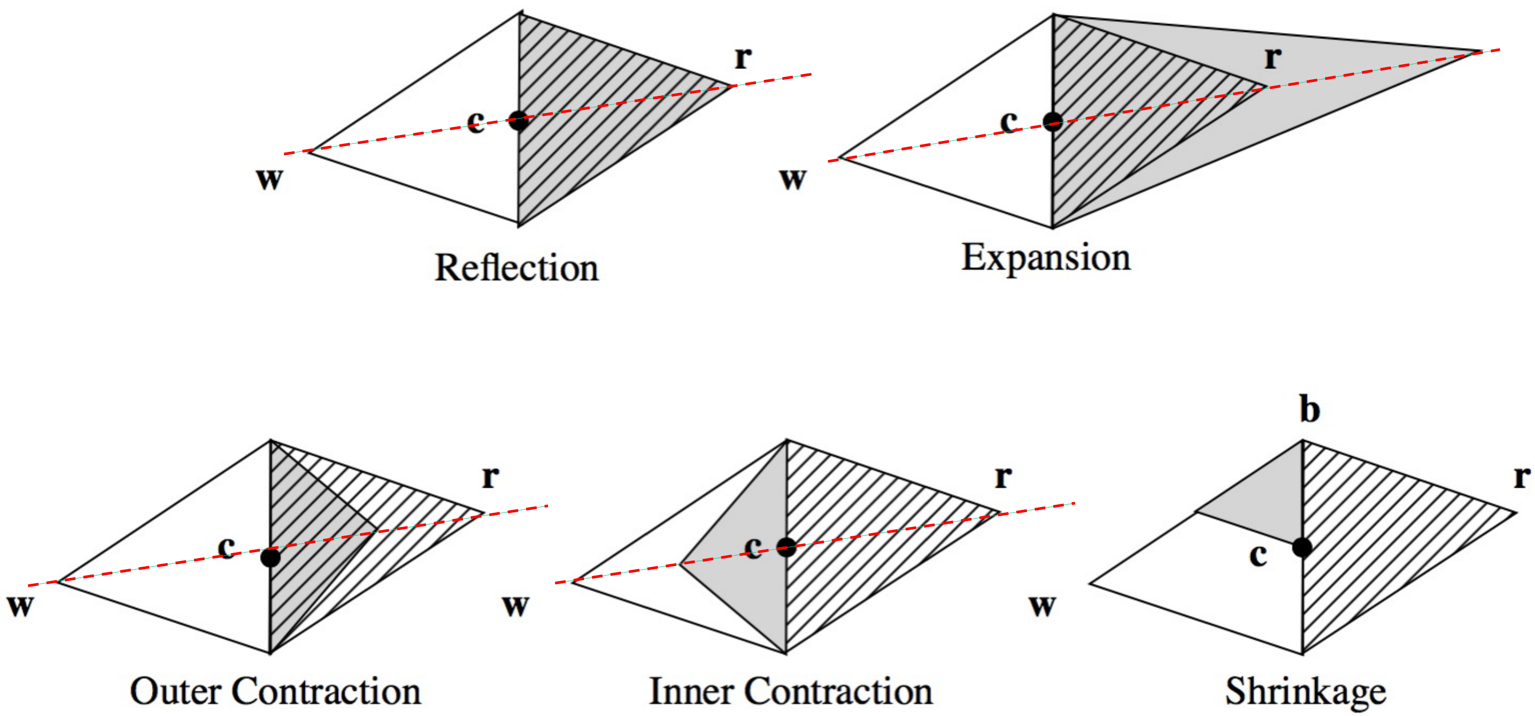
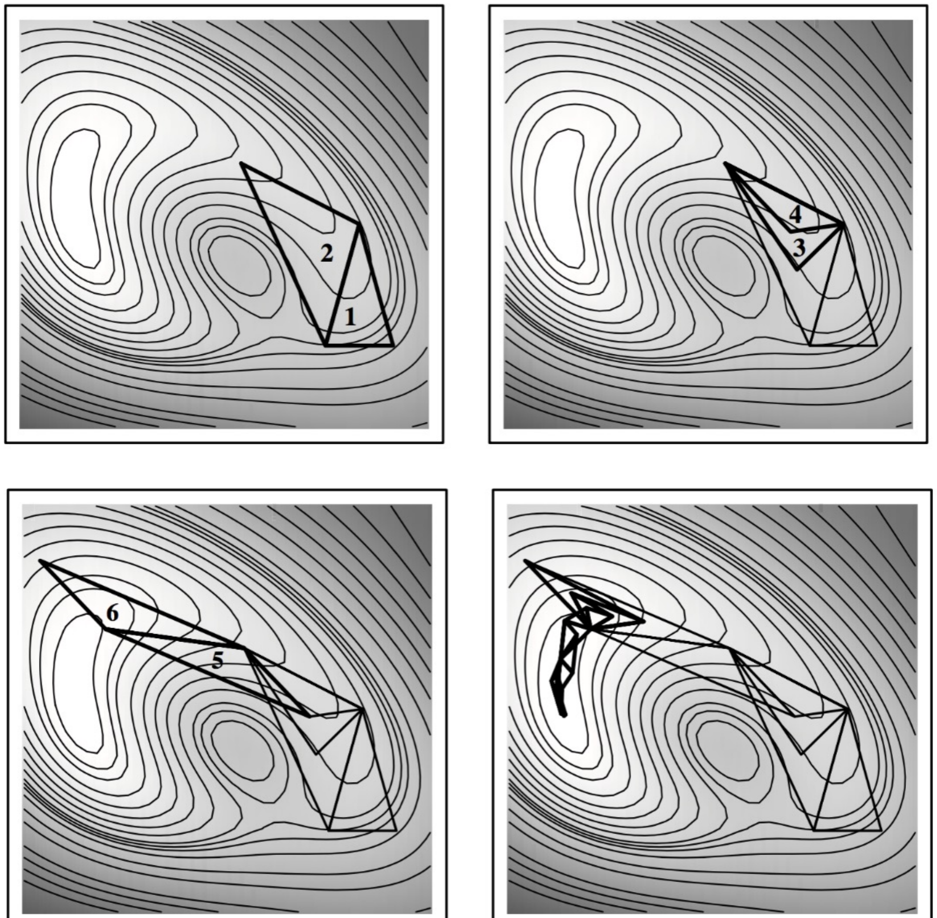


Fig. 2.10 Five possible transformation of a simplex

Key Steps of Nelder–Mead



Practical Note

- Initialization:
  - ➢ start from a simplex around an initial guess for the optimum
- Stopping:
  - ➢ a lack of change in $\mathbf{x}_{\text{best}}$ alone will not be sufficient to stop
  - ➢ $\mathbf{x}_{\text{best}}$ may remain unchanged for several successive iterations
  - ➢ monitor the simplex volume instead
- Performance:
  - ➢ may fail sometimes
  - ➢ convergence speed is relatively slow

Failure & Redeems

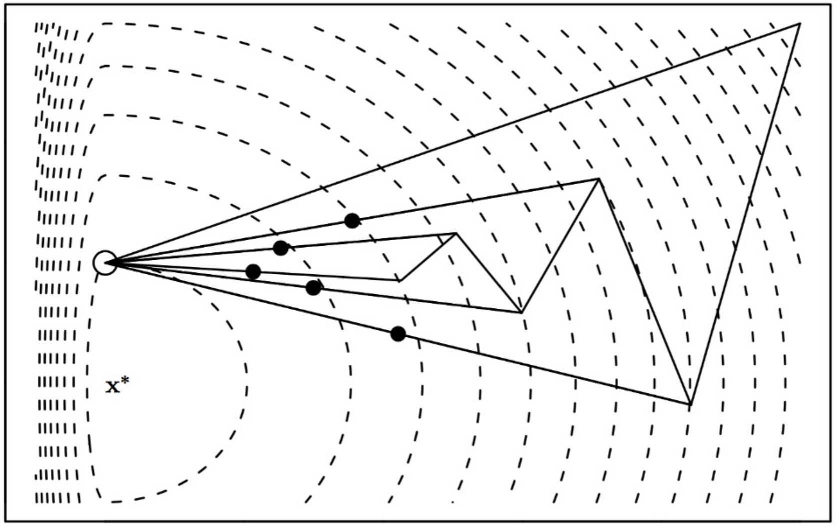


Fig. 2.13 An example for which M_1' fails to find the optimum

Redeems

- Restart with a different simplex
- Oriented restart: reshape the simplex in a manner targeting steepest ascent

Practical Notes

- ➢ M_1' works well for low to moderate dimensions; for high-dimensional problems, its effectiveness is more varied, depending on the nature of the problem
- ➢ M_1' is quite robust for a wide range of functions and random noise
- ➢ It can be implemented with great numerical efficiency, and is a very good candidate for many optimization problems

Reference

- Reference: [T1] Chapter 2.2
- Further reading:
  - [R1] Chapter 6: Solution of Nonlinear Equations & Optimization