Chapter 2: Optimization and Solving Nonlinear Equations

Presented by: Weiyu Li, Mingliang Ma

2018.03
Focus

1. Introduction

2. Univariate Problems
Bisection, Newton’s method, fixed-point iteration

3. Multivariate Problems
Newton’s method, Newton-like methods, Gauss-Newton method, Nelder-Mead algorithm, nonlinear Gauss-Seidel iteration
Example 1:

Maximize

\[ g(x) = \frac{\log x}{1 + x} \]  \hspace{1cm} (1)

with respect to \( x \).

We cannot find the root of \( g'(x) = \frac{1 + \frac{1}{x} - \log x}{(1 + x)^2} \) analytically.

![Figure 1: Illustration of \( g'(x) \). We can make an impression on how \( g'(x) \) hits the x-axis.](image)
Example 2: (Maximum Likelihood Estimation, MLE)

- $l$: the log likelihood function
- $\theta$: the corresponding parameter vector
- $\hat{\theta}$: MLE, maximizes $l$
- $\hat{\theta}$ is the solution of score equation

\[
l'(\theta) = 0 \tag{2}
\]

Finding a MLE amounts to finding a root of the score equation.
Introduction: aims

- **Optimization**: maximization/minimization
- **Solving equations**: non-linear/with constraints (discussed in chap3)/...

Optimization is intimately linked with solving nonlinear equations.

(Conversely, one may also turn a univariate root-finding exercise $g(x) = 0$ into an optimization problem by minimizing $|g(x)|$ with respect to $x$.)
Introduction: iterations

• *initial guess/starting value:* \( x^{(0)} \)

• *updating equation:* produce \( x^{(t+1)} \) from \( x^{(t)} \), \( t = 0, 1, 2, \ldots \)

• *stopping rule:* convergence/no or few development/time...

If \( x^{(t+1)} \) behave poor, discard it and restart in a revised manner.
Introduction: convergence

Convergence Criteria:

• **absolute convergence criterion:**

\[ |x^{(t+1)} - x^{(t)}| < \epsilon \] (3)

• **relative convergence criterion:**

\[ \frac{|x^{(t+1)} - x^{(t)}|}{|x^{(t)}|} < \epsilon, \quad \text{or} \quad \frac{|x^{(t+1)} - x^{(t)}|}{|x^{(t)}| + \epsilon} < \epsilon \] (4)
Convergence Order:
Define $\epsilon(t) = x(t) - x^*$, where $x^*$ is the true solution.

A method has *convergence of order* $\beta$ if $\lim_{t \to \infty} \epsilon(t) = 0$ and

$$
\lim_{t \to \infty} \frac{\epsilon(t+1)}{\epsilon(t)^\beta} = c
$$

for some constants $c \neq 0$ and $\beta > 0$.

Orders v.s. Robustness:
With higher orders of convergence, the precise approximation are more quickly achieved at the expense of robustness; some slow algorithms are more foolproof than their faster counterparts.
Iterative Methods in Univariate Problems

1. Bisection Method

2. Newton’s Method
   - Fisher Scoring
   - Secant Method

3. Fixed-Point Iteration
   - Scaling
1. Bisection Method: basis

Aim
find a root of

\[ g'(x) = 0 \] (6)

Intermediate Value Theorem
If \( g' \) is continuous on \([a_0, b_0]\) and \( g'(a_0)g'(b_0) \leq 0 \), then there exists at least one \( x^* \in [a_0, b_0] \) for which \( g'(x^*) = 0 \) and hence \( x^* \) is a local optimum of \( g \).

Nested Intervals Theorem
If the intervals \( I_n = [a_n, b_n], \ n = 0, 1, 2, \ldots \) satisfy \( [a_0, b_0] \supset [a_1, b_1] \supset [a_2, b_2] \supset \cdots \), then the intersection of the \( I_n \) is not empty.
Moreover, if the length of the intervals converges to 0, then \( a_n, b_n \) converge to the same singleton.
1. Bisection Method: algorithm

- **starting values:** \([a_0, b_0]\) satisfying \(g'(a_0)g'(b_0) \leq 0\), \(x^{(0)} = (a_0 + b_0)/2\).

- **updating equations:**

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

  \begin{equation}
  \text{(7)}
  \end{equation}

  and

  \[
  x^{(t+1)} = \frac{(a_{t+1} + b_{t+1})}{2}.
  \]

  \begin{equation}
  \text{(8)}
  \end{equation}

- **stopping rules:** (example)

  stop after \(t > \log_2\{(b_0 - a_0)/\delta\} - 1\) iterations, if true error tolerance \(|x^{(t)} - x^*| < \delta\).
1. Bisection Method: remarks

- smoothness required: continuity of $g'$
- more than one root in $[a_0, b_0]$
- $a_{t+1} = a_t + (b_t - a_t)/2$
- bracketing method: bounding a root within a sequence of nested intervals of decreasing length
- slow approach

The distances $x^{(t)} - x^*$ need not shrink at every iteration and their ratio is potentially unbounded. Thus bisection does not formally meet the definition for determining order of convergence $\beta$. 
1. Bisection Method: example

Find the value of $x$ maximizing

$$g(x) = \frac{\log\{x\}}{1 + x}$$

We take $a_0 = 1$, $b_0 = 5$, and $x^{(0)} = 3$. Figure 2 illustrates the first few steps of the bisection algorithm.
Figure 2: Illustration of the bisection method. The top portion of this graph shows $g'(x)$ and its root at $x^*$. The bottom portion shows the first three intervals obtained using the bisection method with $(a_0, b_0) = (1, 5)$. The $t$th estimate of the root is at the center of the $t$th interval.
2. Newton’s Method

- Supposition: $g'$ continuously differentiable and $g''(x^*) \neq 0$

- linear Taylor series expansion:

$$0 = g'(x^*) \approx g'(x^{(t)}) + (x^* - x^{(t)})g''(x^{(t)}).$$  \hspace{1cm} (10)

Thus,

- updating equation:

$$x^* \equiv x^{(t+1)} = x^{(t)} - \frac{g'(x^{(t)})}{g''(x^{(t)})} \overset{\text{def}}{=} x^{(t)} + h^{(t)}$$ \hspace{1cm} (11)

(In an MLE problem, $\theta^{(t+1)} = \theta^{(t)} - \frac{\theta'(t)}{\theta''(t)}$.)
2. Newton’s Method: example

For maximizing \( g(x) = \frac{\log x}{1+x} \),

\[
h^{(t)} = \frac{(x^{(t)} + 1)(1 + 1/x^{(t)} - \log\{x^{(t)}\})}{3 + 4/x^{(t)} + 1/(x^{(t)})^2 - 2 \log\{x^{(t)}\}}.
\]

(12)

We start from \( x^{(0)} = 3.0 \). Figure 3 illustrates the first several iterations. Newton’s method quickly finds \( x^{(4)} \approx 3.59112 \). For comparison, the first five decimal places of \( x^* \) are not correctly determined by the bisection method until iteration 19.

Table 1: Values of \( x^{(t)} \) using Newton’s method. It is obvious to notice the quick convergence of the solution.

<table>
<thead>
<tr>
<th>Iteration, ( t )</th>
<th>( x^{(t)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.000000</td>
</tr>
<tr>
<td>1</td>
<td>3.417789</td>
</tr>
<tr>
<td>2</td>
<td>3.574045</td>
</tr>
<tr>
<td>3</td>
<td>3.590946</td>
</tr>
<tr>
<td>4</td>
<td>3.591121</td>
</tr>
<tr>
<td>5</td>
<td>3.591121</td>
</tr>
</tbody>
</table>
Figure 3: Illustration of Newton’s method applied to maximize $g(x) = \frac{\log x}{1+x}$. At the first step, Newton’s method approximates $g'$ by its tangent line at $x^{(0)}$ whose root, $x^{(1)}$, serves as the next approximation of the true root, $x^*$. The next step similarly yields $x^{(2)}$, which is already quite close to the root at $x^*$.
2. Newton’s Method: convergence

- Convergence depends on the shape of $g$ and the starting value.

**Theorem.** If $g'''$ is continuous and $x^*$ is a simple root of $g'$, then there exists a neighborhood of $x^*$ for which Newton’s method converges to $x^*$ when started from any $x^{(0)}$ in that neighborhood.

**Proof.** Since $x^*$ is a simple root, there exists a neighborhood of $x^*$ within which $g''(x) \neq 0$ for all $x$. Consider in this neighborhood. Define $\epsilon^{(t)} = x^{(t)} - x^*$.

From Taylor expansion,

$$0 = g'(x^*) = g'(x^{(t)}) + (x^* - x^{(t)})g''(x^{(t)}) + \frac{1}{2}(x^* - x^{(t)})^2 g'''(q)$$

for some $q$ between $x^*$ and $x^{(t)}$. Rearrangement yields

$$\epsilon^{(t+1)} = (\epsilon^{(t)})^2 \frac{g'''(q)}{2g''(x^{(t)})}.$$
Let
\[ c(\delta) = \max_{x_1, x_2 \in N_\delta(x^*)} \left| \frac{g'''(x_1)}{2g''(x_2)} \right| \rightarrow \frac{g'''(x^*)}{2g''(x^*)} \] (15)
as \( \delta \rightarrow 0 \). Choose \( \delta \) such that \( \delta c(\delta) < 1 \). Then,
\[ |\epsilon(t)| \leq \frac{(c(\delta)\delta)^{2t}}{c(\delta)} \rightarrow 0 \] (16)
as \( t \rightarrow \infty \). Hence \( x^{(t)} \rightarrow x^* \). □

We also find that:
Newton’s method has quadratic convergence order, \( \beta = 2 \).
2. Newton’s Method: relationship with bisection

- Bisection safeguard an interval within which a root must lie.

**Example:**

Figure 4: Starting from $x^{(0)}$, Newton’s method diverges by taking steps that are increasingly distant from the true root, $x^*$. However, bisection would have found this root easily.
2. Newton’s Method: Fisher scoring

- **approximation of** \( l''(\theta^{(t)}) \)

- **Fisher information** \( I(\theta) = E\{l'(\theta)^2\} = -E\{l''(\theta)\} \)

- **updating equation:**
  \[
  \theta^{(t+1)} = \theta^{(t)} + l'(\theta^{(t)})I(\theta^{(t)})^{-1}
  \]  
  \( (17) \)

- \( \beta \approx 2 \)

Fisher scoring and Newton’s method share the same asymptotic properties; either may be easier for a particular problem. Generally, FS makes rapid improvements initially, while NM gives better refinements near the end.
2. Newton’s Method: secant method

• **approximation:** \[ g''(x^{(t)}) \approx \frac{g'(x^{(t)}) - g'(x^{(t-1)})}{x^{(t)} - x^{(t-1)}} \]

• **updating equation:**

\[
x^{(t+1)} = x^{(t)} - \frac{x^{(t)} - x^{(t-1)}}{g'(x^{(t)}) - g'(x^{(t-1)})} \tag{18}
\]

for \( t \geq 1 \). Requires two starting points, \( x^{(0)} \) and \( x^{(1)} \).

• \( \beta = \frac{1+\sqrt{5}}{2} \approx 1.62 \)

Order of convergence is superlinear, but slower than Newton’s method.
Figure 5: The secant method locally approximates $g'$ using the secant line between $x^{(0)}$ and $x^{(1)}$. The corresponding estimated root, $x^{(2)}$, is used with $x^{(1)}$ to generate the next approximation.
3. Fixed-Point Iteration: contraction mapping

- **fixed point**: a point whose evaluation by that function equals itself

- **contractive mapping** $G$ on $[a, b]$: if

  1. $G([a, b]) \subset [a, b]$
  2. **Lip condition with Lip constant** $\lambda \in [0,1)$:
     \[ |G(x_1) - G(x_2)| \leq \lambda |x_1 - x_2| \text{ for all } x_1, x_2 \in [a, b] \]

**Contractive Mapping Theorem**

If $G$ is a contractive mapping, then there exists a unique fixed point $x^*$ in this interval, and $G^n(x) \to x^*$ as $n \to \infty$ for all $x \in [a, b]$. 
3. Fixed-Point Iteration: method

• find contractive mapping $G(x)$, s.t. $g'(x) = 0 \iff G(x) = x$

• updating equation:

$$x^{(t+1)} = G(x^{(t)})$$

(19)

• precision: $|x^{(t)} - x^*| \leq \frac{\lambda^t}{1-\lambda} |x^{(1)} - x^{(0)}|$

• choice of $G(x)$:

- $G(x) = g'(x) + x \implies x^{(t+1)} = x^{(t)} + g'(x^{(t)})$
- $G(x) = x - \frac{g'(x)}{g''(x)} \implies$ Newton’s Method
- $G(x) = \alpha g'(x) + x \implies$ Scaling
3. Fixed-Point Iteration: scaling

We can often fix non-contractive functions by rescaling $g$ or $G$.

**set** $G(x) = \alpha g'(x) + x$ **for a suitable** $\alpha$. **Find** $\alpha$ **by trial and error or analysis of** $g''$. **For example**, choose $\alpha$ **satisfying** $|\alpha g''(x) + 1| < 1$ **for all** $x \in [a, b]$.

An example using $\alpha = 4$ is showed on the following figure.

Effectiveness of fixed point iteration also depends strongly on the chosen functional form of $G$.

For example, consider finding the root of $g'(x) = x + \log x$. Then $G(x) = (x + e^{-x})/2$ converges quickly, whereas $G(x) = e^{-x}$ converges more slowly and $G(x) = -\log x$ fails to converge at all.
Figure 6: The first three steps of scaled fixed point iteration to maximize $g(x) = \frac{\log(x)}{1+x}$ using $G(x) = g'(x) + x$ and scaling with $\alpha = 4$. 
3. Fixed-Point Iteration: remarks

- compare with Newton method:

Suppose $l$ is (nearly) quadratic. Then $l'$ is (nearly) locally linear, and $l''$ is roughly a constant, say $\gamma$. For a quadratic log likelihood, Newton’s method would use the updating equation $\theta^{(t+1)} = \theta^{(t)} - l'(\theta)/\gamma$. If we use scaled fixed point iteration with $\alpha = -1/\gamma$, we get the same updating equation!

Many log likelihoods are approximately locally quadratic. Thus, scaled fixed point iteration can be very effective.

- F-P is also stable and easy to code.
Iterative Methods in Multivariate Problems

0. Review

1. Newton-like Methods
   - Ascent Algorithms and Backtracking
   - Discrete Newton and Fixed-Point Methods
   - Quasi-Newton Methods

2. Gauss-Newton Method
   - *Iteratively Reweighted Least Squares

3. Nelder-Mead Algorithm

4. Nonlinear Gauss-Seidel Iteration
0. Review: Newton’s method and fisher scoring

Aim

$g : \mathbb{R}^p \rightarrow \mathbb{R}^1$, find a root of $g'(x) = 0$.

Derivations

1. approximate $g'$ by linear Taylor expansion
2. approximate $g$ by quadratic, then set the gradient to zero

Both yields

$$g'(x^{(t)}) + g''(x^{(t)})(x^* - x^{(t)}) = 0. \tag{20}$$

Newton’s method

$$x^{(t+1)} = x^{(t)} - g''(x^{(t)})^{-1}g'(x^{(t)}). \tag{21}$$

Fisher scoring

$$\theta^{(t+1)} = \theta^{(t)} + I(\theta^{(t)})^{-1}l'(\theta^{(t)}). \tag{22}$$
A failure in Newton’s method

Figure 7: An application of Newton’s method for maximizing a complex bivariate function. The surface of the function is indicated by shading and contours, with light shading corresponding to high values. Two runs starting from $z_a^{(0)}$ and $z_b^{(0)}$ are shown. These converge to the true maximum and to a local minimum, respectively.

Newton’s method is not guaranteed to walk *uphill*. It is not guaranteed to find a local *maximum*. Step length matters even when step direction is good.
1. Newton-like methods

- form:

\[ x^{(t+1)} = x^{(t)} - (M^{(t)})^{-1}g'(x^{(t)}) \]  \hspace{1cm} (23)

where \( M^{(t)} \) is a \( p \times p \) matrix approximating the Hessian, \( g''(x^{(t)}) \).

- advantages:

  - computationally easy: avoid calculating Hessian
  - guarantee uphill steps (ascent)
1. Newton-like methods: choice of $M^{(t)}$

- **Newton’s method:** $g''(x^{(t)})$
- **Fisher scoring:** $-I(\theta^{(t)})$
- **Ascent algorithms:** $(\alpha^{(t)})^{-1}M^{(t)}$, where $M^{(t)} < 0$ and $\alpha^{(t)} \to 0$ to ensure increment
- **Steepest ascent:** $-I$, thus $x^{(t+1)} = x^{(t)} + g'(x^{(t)})$
- **Multivariate fixed-point methods:** constant $M$ (reasonably choose $g''(x^{(0)})$)
- **Discrete Newton method:** $M_{ij}^{(t)} = \frac{g'_i\left(x^{(t)}+h_{ij}^{(t)}e_j\right)-g'_i\left(x^{(t)}\right)}{h_{ij}^{(t)}}$
- **Quasi-Newton methods:** updating $M^{(t+1)}$ to $M^{(t)}$
1. Newton-like methods: backtracking

- If downhill, change the contraction/step length parameter $\alpha(t)$ to the uphill direction.

- e.g. step halving: $\alpha(t) \leftarrow \alpha(t)/2$

Why $\alpha(t)$ and $M(t) < 0$?

Let updating increment $h(t) = -\alpha(t) [M(t)]^{-1}g'(x(t))$, and fix $x(t)$ and $M(t) < 0$, then as $\alpha(t) \to 0$ we have

$$
g(x^{(t+1)}) - g(x^{(t)}) = g(x^{(t)} + h^{(t)}) - g(x^{(t)})
= -\alpha(t)g'(x^{(t)})^T(M^{(t)})^{-1}g'(x^{(t)}) + \sigma(\alpha(t)), \tag{24}
$$

Therefore ascent can be ensured by choosing $\alpha(t)$ sufficiently small, yielding

$$
g(x^{(t+1)}) - g(x^{(t)}) > 0
$$

from (24) since $\sigma(\alpha(t))/\alpha(t) \to 0$ as $\alpha(t) \to 0$. 
1. Newton-like methods: more about backtracking

Steepest ascent: $-M^{(t)} = I > 0$, guarantees uphill steps.
Fisher scoring: $-M^{(t)} = I(\theta^{(t)}) \geq 0$, avoids stepping downhill.

- **line search methods**: finding an advantageous length in the chosen direction, including backtracking.

- **modified Newton methods**: e.g. Cholesky decomposition approach to make negative definite matrix.
The steepest ascent direction is not necessarily the wisest, and backtracking doesn’t prevent oversteps. The example will be compared with quasi-Newton method later.
1. Newton-like methods: discrete Newton method

- approximation of the Hessian
- analogy of secant method in univariate problems:
  \[ M_{ij}^{(t)} = \frac{g'_i(x^{(t)} + h_{ij}^{(t)}e_j) - g'_i(x^{(t)})}{h_{ij}^{(t)}} \]  

  where \( g'_i(x) = \frac{dg(x)}{dx_i} \) is the \( i \)th element of \( g'(x) \), \( e_j \) is the \( p \)-vector with a 1 in the \( j \)th position and zeros elsewhere, and \( h_{ij}^{(t)} \) are some constants.

- choice of \( h_{ij}^{(t)} \):
  \( h_{ij}^{(t)} = h \) for all \( (i,j) \) and \( t \) leads to linear convergence order: \( \beta = 1 \).
  Alternatively, \( h_{ij}^{(t)} = x_j^{(t)} - x_j^{(t-1)} \) for all \( i \) gives superlinear convergence.

- remark: ensure symmetry by calculating \( \frac{M^{(t)} + (M^{(t)})'}{2} \)
1. Newton-like methods: quasi-Newton methods

- updating based on the recent step, not wholly recalculating each step

- secant condition:

\[ g'(x^{(t+1)}) - g'(x^{(t)}) = M^{(t+1)}(x^{(t+1)} - x^{(t)}). \]

(26)

Goal: generate \( M^{(t+1)} \) from \( M^{(t)} \) in a manner that requires few calculations and satisfies (26), while learning about the curvature of \( g' \) in the direction of the most recent step.
1. Newton-like methods: some updates in quasi-Newton

1. symmetric rank-one update

Let \( z^{(t)} = x^{(t+1)} - x^{(t)} \) and \( y^{(t)} = g'(x^{(t+1)}) - g'(x^{(t)}) \). Then we can write the update to \( M^{(t)} \) as

\[
M^{(t+1)} = M^{(t)} + c^{(t)} v^{(t)} (v^{(t)})^T
\]

where \( v^{(t)} = y^{(t)} - M^{(t)} z^{(t)} \) and \( c^{(t)} = \frac{1}{(v^{(t)})^T z^{(t)}} \).

2. a rank-two update: BFGS update

\[
M^{(t+1)} = M^{(t)} - \frac{M^{(t)} z^{(t)} (M^{(t)} z^{(t)})^T}{(z^{(t)})^T M^{(t)} z^{(t)}} + \frac{y^{(t)} (y^{(t)})^T}{(z^{(t)})^T y^{(t)}}
\]

- Update1 does not confer hereditary positive definiteness; BFGS does.
- There are still controversies of which one performs better.
- Performance sensitive to choice of \( M^{(0)} \)
Convergence of quasi-Newton methods is generally superlinear, but not quadratic. These are powerful and popular methods, available, for example, in the R function `optim()`.
1. Newton-like methods: *remarks about the Hessian

- reapproximate after iterations
- central difference approximation

\[ h^{(t)}_{ij} = h = \epsilon^{1/3} \], where \( \epsilon \) represents the computer’s floating-point precision
2. Gauss-Newton method: aim

For nonlinear least squares problems with observed data \((y_i, z_i)\) for \(i = 1, \ldots, n\) and model

\[
Y_i = f(z_i, \theta) + \epsilon_i
\]  

(29)

for some non-linear function, \(f\), and random error, \(\epsilon_i\).

We seek to estimate \(\theta\) by maximizing an objective function

\[
g(\theta) = -\sum_{i=1}^{n} (y_i - f(z_i, \theta))^2.
\]  

(30)
2. Gauss-Newton method: regression!

By linear Taylor series expansion, we have

\[ Y_i \approx f(z_i, \theta^{(t)}) + (\theta - \theta^{(t)})^T f'(z_i, \theta^{(t)}) + \epsilon_i. \]  

(31)

Let \( X_i^{(t)} \) denote a working response whose observed value is

\[ x_i^{(t)} = y_i - f(z_i, \theta^{(t)}), \]

and define \( a_i^{(t)} = f'(z_i, \theta^{(t)}) \). With rearrangement, the maximization of \( g \) can be re-expressed as minimizing the squared residuals of the linear regression model

\[ X^{(t)} = A^{(t)}(\theta - \theta^{(t)}) + \epsilon \]  

(32)

This is a regression problem! Thus we have the updating equation:

\[ \theta^{(t+1)} = \theta^{(t)} + \left((A^{(t)})^T A^{(t)}\right)^{-1}(A^{(t)})^T x^{(t)}. \]  

(33)

Requires no computation of Hessian.
Works best when the model fits fairly well and \( f \) is not severely nonlinear.
2. Gauss-Newton method: *Iterative Reweighted Least Squares

- Find the MLEs for the parameters of a logistic regression model, or other GLM models.
- Model the response variables as $Y_i|z_i \sim Bernoulli(\pi_i)$ independently.
- Define $z_i = (1, z_i)^T$ and $\beta = (\beta_0, \beta_1)^T$.
- Exponential family form: $f(y|\theta) = e^{[y\theta-b(\theta)]/a(\phi)+c(y,\phi)}$

With some calculation, we conclude that

$$l(\beta) = y^T Z\beta - b^T 1, \quad (34)$$
$$l'(\beta) = Z^T (y - \pi), \quad (35)$$
$$l''(\beta) = -Z^T W Z, \quad (36)$$

where $W = \text{diag}(\pi_1(1 - \pi_1), \ldots, \pi_n(1 - \pi_n))$.

Therefore, the updating function is

$$\beta^{(t+1)} = \beta^{(t)} + (Z^T W^{(t)} Z)^{-1} \left(Z^T (y - \pi^{(t)})\right), \quad (37)$$

where $\pi^{(t)}$ is the value of $\pi$ corresponding to $\beta^{(t)}$, and $W^{(t)}$ is the diagonal weight matrix evaluated at $\pi^{(t)}$. 
3. Nelder-Mead Algorithm: outline

- neighbourhood: approximating the maximum
- reshape/resize: a new one replace the worst

Direct search approach: depends only on a collection of evaluations at possible solutions, and nominate a superior point for the next iteration. Including genetic algorithms and simulated annealing in chap3.
3. Nelder-Mead Algorithm: notations

- p-dimensional simplex $x_1, \ldots, x_{p+1}$

- $g(x_1) \geq \ldots \geq g(x_{p+1})$

- $x_{best}, x_{worst}, x_{bad}$

- the best face, $c = \frac{1}{p} \left[ \left( \sum_{i=1}^{p+1} x_i - x_{worst} \right) \right]$
3. Nelder-Mead Algorithm: replacement

- **search direction:** $c - x_{\text{worst}}$

- **reflection:** $x_r = c + \alpha_r (c - x_{\text{worst}})$, $\alpha_r > 0$ (*usually* $\alpha_r = 1$)

  If $g(x_r) > g(x_{\text{best}})$,

- **expansion:** $x_e = c + \alpha_e (x_r - c)$, $\alpha_e > \max(1, \alpha_r)$ (*usually* $\alpha_e = 2$)

  If $g(x_{\text{best}}) \geq g(x_r) > g(x_{\text{bad}})$, then $x_r$.

  If $g(x_{\text{bad}}) \geq g(x_r) > g(x_{\text{worst}})$,

- **outer contraction:** $x_o = c + \alpha_c (x_r - c)$, $a < \alpha_c < 1$ (*usually* $\alpha_c = \frac{1}{2}$)

  If $g(x_r) \leq g(x_{\text{worst}})$,

- **inner contraction:** $x_i = c + \alpha_c (x_{\text{worst}} - c)$

  If no improvement in situation,

- **shrink transformation:**
  
  $x_{sj} = x_{\text{best}} + \alpha_s (x_j - x_{\text{best}})$, $a < \alpha_s < 1$ (*usually* $\alpha_s = \frac{1}{2}$) $c = \frac{1}{p}[(\sum_{i=1}^{p+1} x_i - x_{\text{worst}})]$
3. Nelder-Mead Algorithm: summary

1. Initialize.

2. Sort $x_{\text{best}}, x_{\text{worst}}, x_{\text{bad}}$.

3. Reshape.

4. Stop or 2.

Figure 10: The five possible transformations of a simplex.
3. Nelder-Mead Algorithm: example

Figure 11: Nelder-Mead algorithm for maximizing a complicated bivariate function.
3. Nelder-Mead Algorithm: remarks

- convergence criteria
  1. convergence of simplex volume
  2. convergence of the function values

- high-dimensional problems

- robustness: wide-ranging functions and noise

- efficiency: always 2 values, seldom p values for shrinkage

- relatively slow

- rare failure!
3. Nelder-Mead Algorithm: failure and revision

Figure 12: Contours of a function and successive simplicies. $x^*$ is the global maximum. It shows a failure of Nelder-Mead algorithm. The simplices converge to the point that is neither local maxima nor minima.

- **oriented restart**
4. Nonlinear Gauss-Seidel iteration

- *backfitting / cyclic coordinate ascent*

- view $g'_j$ as a univariate real function of $x_j$, i.e. $g'_j = g'_j(x_j)$

- cycle through all $p$ components using the most recent values

- advantages:
  1. simplifies problems
  2. easy and stable to compute
  3. quick

- applications: fitting additive models, GAMs, etc.
Figure 13: Gauss-Seidel iteration with univariate quasi-Newton sub-steps. Each line segment represents a change of a single coordinate in the current solution, so complete steps from $x^{(t)}$ to $x^{(t+1)}$ correspond to pairs of adjacent segments.

Note that the very first univariate optimization (one horizontal step left from $x^{(0)}$) actually failed, finding a local univariate minimum instead of the global univariate maximum. Not advised, but subsequent Gauss-Seidel iterations overcame this misstep!
Coordinate descent:

\[ x_i^{(t+1)} = \arg\min_{y \in \mathbb{R}} f(x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, y, x_{i+1}^{(t)}, \ldots, x_n^{(t)}) \]  

\[ \text{(38)} \]

Batch/stochastic/mini-batch gradient descent

Hard thresholding: \( \arg\min_x \|Ax - b\|_2^2 + \lambda \|x\|_0 \)

\[ \eta_H(w, \lambda) = w I_{|w| > \lambda} \]  

\[ \text{(39)} \]

Soft thresholding: \( \arg\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1 \)

\[ \eta_S(w, \lambda) = \text{sgn}(w) (|w| - \lambda)_+ \]  

\[ \text{(40)} \]
Figure 14: Illustration of the hard and soft thresholding functions. via the Internet.