Lecture 1
Root-finding, Optimization, Numerical Integration

Peifan Wu

UBC

Root-finding

Root-finding Problem

- We are interested in finding x^* that satisfies

$$f(x) = 0 \quad f : \mathbb{R} \to \mathbb{R}$$

  Or the fixed point problem

$$f(x) = x$$

- Methods:
  - Bisection method
  - Newton's method
  - Quasi-Newton and Broyden's algorithm

Bisection Method

- Simple and robust for one-dimensional continuous function on a closed interval

- Suppose $f(x)$ is defined on $[a, b]$ where $f(a)$ and $f(b)$ have opposite signs
- e.g., $f(a) < 0$ and $f(b) > 0$

- By the mean value theorem there exists at least a zero

  1. Set $n = 1$, $a^n = a$, and $b^n = b$
  2. Compute $c^n = \frac{a^n + b^n}{2}$
  3. If $f(c^n) < 0$ set $a^{n+1} = c^n$ and $b^{n+1} = b^n$. Otherwise set $b^{n+1} = c^n$ and $a^{n+1} = a^n$
  4. Go back to step 2. until convergence criterion is satisfied

Advantages and Disadvantages of Bisection Method

- Robust and Stable: guaranteed to find an approximate solution
- Does not require computation of derivatives: needs continuity but not differentiability
- Slow because it does not exploit information on the slope of the function
- Cannot be generalized to the multi-dimensional case

Newton's Method

- Approximate the function f at point x^n in iteration n as

$$f(x) \approx f(x^n) + f'(x^n)(x - x^n)$$

  and update x as if you were solving for a zero of the approximation

$$x^{n+1} = x^n - \frac{f(x^n)}{f'(x^n)}$$

- f must be differentiable
- fast, but unstable if function changes derivative quickly

Quasi-Newton Method

- Use the slope of the secant function going through $f\left(x^n\right)$ and $f\left(x^{n+1}\right)$

- Iteration scheme becomes

$$x^{n+1} = x^n - \left[\frac{x^n - x^{n-1}}{f\left(x^n\right) - f\left(x^{n-1}\right)}\right] f\left(x^n\right)$$

- convergence is slower than Newton method

Broyden's Algorithm

- Multidimensional version of the univariate quasi-Newton
- Let $\mathbf{f} : \mathbb{R}^k \to \mathbb{R}^k$, a system of k equations and k unknowns
- Iteration rule is
$$\mathbf{x}^{n+1} = \mathbf{x}^n - \left(\mathbf{J}^n\right)^{-1} \mathbf{f}\left(\mathbf{x}^n\right)$$
where $\mathbf{J}^n$ is the Jacobian evaluated at $\mathbf{x}^n$
- To avoid computing $\mathbf{J}^n$, use $\mathbf{A}^n$ given by secant method:
$$\mathbf{A}^{n+1} = \mathbf{A}^n + \left[\mathbf{f}\left(\mathbf{x}^{n+1}\right) - \mathbf{f}\left(\mathbf{x}^n\right) - \mathbf{A}^n \mathbf{d}^n\right] \frac{\left(\mathbf{d}^n\right)'}{\left(\mathbf{d}^n\right)' \mathbf{d}^n}$$
where $\mathbf{d}^n = \mathbf{x}^{n+1} - \mathbf{x}^n$
- Guess of $\mathbf{A}^0$: scaled identity matrix

Optimization

Minimization

- We describe minimization problems: to maximize f is to minimize $-f$
- Isomorphisms:
  - If f is concave then the minimization problem

    $$\min_{x \in [a,b]} f(x)$$

    is equivalent to root-finding on the FOC $f'(x) = 0$
  - The multivariate root-finding problem

    $$\begin{aligned} f^1(x_1, x_2, \ldots, x_n) &= 0 \\ f^2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f^n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

    can be restated as a nonlinear least square problem

    $$\min_{\{x_1, x_2, \ldots, x_n\}} \sum_{i=1}^{n} f^i(x_1, x_2, \ldots, x_n)^2$$

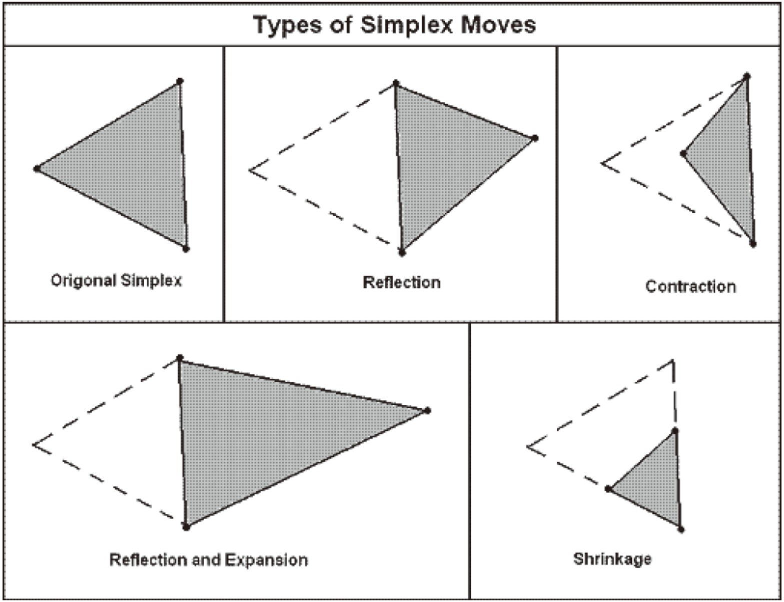
Bracketing Method

- Most reliable for one dimensional problems
- Initialization: find $a < b < c$ such that $f(a), f(c) > f(b)$
  1. Choose $d \in (a, b)$ and compute $f(d)$
  2. Choose new (a, b, c) triplet:
     If $f(d) > f(b)$ then the minimum is in $[d, c]$. Update the triple (a, b, c) with (d, b, c).
     If $f(d) < f(b)$ then the minimum is in $[a, b]$. Update the triple (a, b, c) with (a, d, b).
  3. Stop if $c - a < \delta$. If not, got back to step 1.
- Golden search: Choose the segment interval following Golden ratio.

Simplex Method

- Multidimensional comparison method, also called Nelder-Meade or polytope method.
- In Matlab: fminsearch

1. Choose initial simplex $\{x_1, x_2, \ldots, x_n, x_{n+1}\} \in \mathbb{R}^n$
2. Reorder the simplex vertices in descending order: $f(x_i) \geqslant f(x_{i+1}), \forall i$
3. Find the smallest i such that $f\left(x_i^R\right) < f(x_i)$ where x_i^R is the reflection of x_i. If exists, replace x_i with x_i^R and go back to step 2.
4. If width of the current simplex is $< \epsilon$, stop
5. For $i = 1, \ldots, n$ set $x_i^S = \frac{x_i + x_{i+1}}{2}$ to shrink the simplex. Go back to step 1.


Types of Simplex Moves
Origonal Simplex
Reflection
Contraction
Reflection and Expansion
Shrinkage

Newton / Quasi-Newton Methods

– Second-order Taylor approximation of f yields

$$f\left(\mathbf{x}^{n+1}\right) \approx f\left(\mathbf{x}^{n}\right) + \nabla f\left(\mathbf{x}^{n}\right)'\left(\mathbf{x}^{n+1} - \mathbf{x}^{n}\right) + \frac{1}{2}\left(\mathbf{x}^{n+1} - \mathbf{x}^{n}\right)' H\left(\mathbf{x}^{n}\right)\left(\mathbf{x}^{n+1} - \mathbf{x}^{n}\right)$$

– Updating equation in $\mathbb{R}^n$

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} - H\left(\mathbf{x}^{n}\right)^{-1} \nabla f\left(\mathbf{x}^{n}\right)$$

– Using the updating rule, the Taylor approximation becomes

$$f\left(\mathbf{x}^{n+1}\right) \approx f\left(\mathbf{x}^{n}\right) - \frac{1}{2}\left(\mathbf{x}^{n+1} - \mathbf{x}^{n}\right)' H\left(\mathbf{x}^{n}\right)\left(\mathbf{x}^{n+1} - \mathbf{x}^{n}\right)$$

therefore as long as the Hessian is approximated by a positive definite matrix, the algorithm moves in the right direction

Newton / Quasi-Newton Methods

- Computation of Hessian is time consuming – that's where quasi-Newton methods come handy

- Simplest option: set Hessian to I then

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} - \nabla f\left(\mathbf{x}^{n}\right)$$

  this method is called steepest descent

- Berndt-Hall-Hall-Hausman (BHHH) method uses the outer product of the gradient vectors to replace the Hessian
- Broyden-Fletcher-Foldfarb-Shanno (BFGS) and Davidon-Fletcher-Powell (DFP) algorithms: secant methods that approximate Hessian with symmetric positive definite matrix

Numerical Integration

Numerical Integration Methods

- We are interested in approximating

$$\int_a^b f(x)\, \mathrm{d}x$$

- Three approaches to computing integral:

  1. Newton-Cotes methods: employ piecewise polynomial approximations to the integrand with evenly spaced nodes
  2. Gaussian Quadrature: choose nodes and weights efficiently such that they satisfy some moment-matching conditions
  3. Monte-Carlo methods: use equally weighted random nodes

Newton-Cotes: Trapezoid rule

- Approximate f with piecewise linear function

- Partition the interval $[a, b]$ into n subintervals of equal length $h = (b - a)/n$ and endpoint nodes $x_i = a + ih$

- The area under the piecewise linear approximation for subinterval i is

$$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x \approx \left[\frac{f(x_{i+1}) + f(x_i)}{2}\right] \cdot h$$

  and hence

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{N} \omega_i f(x_i)$$

  where $\omega_i = h$ for all i unless $i = 0, N$ where $\omega_i = h/2$

Newton-Cotes: Simpson rule

- Simpson rule based on quadratic approximation of the function
- Similar expression as trapezoid rule. Figure out what ω_i need to be
- If f is smooth then Simpson rule is preferred because approximation error is square of Trapezoid rule error
- If f is nondifferentiable at some points then trapezoid rule may be better

Gaussian Quadrature

- In general suppose you want to compute an integral of the type

$$\int_a^b f(x)\,\omega(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

- Choose the suitable quadrature

Range	$\omega(x)$	Polynomial family	Quadrature method
$[-1,1]$	1	Legendre	Gauss-Legendre
$(-1,1)$	$\sqrt{1-x^2}$	Chebyshev	Gauss-Chebyshev
$(0,\infty)$	$\exp(-x)$	Laguerre	Gauss-Laguerre
$(-\infty,\infty)$	$\exp\left(-x^2\right)$	Hermite	Gauss-Hermite