Math 273a: Optimization
1D search methods

Instructor: Wotao Yin
Department of Mathematics, UCLA
Fall 2015

based on Chong-Zak, 4th Ed.
Goal

Develop methods for solving the one-dimensional problem

\[
\minimize_{x \in \mathbb{R}} f(x)
\]

under the following cases:

- (0th order info) only objective value \( f \) is available,
- (1st order info) \( f' \) is available, but not \( f'' \),
- (2nd order info) both \( f' \) and \( f'' \) are available.

Higher-order information tends to give more powerful algorithms.

These methods are also used in multi-dimensional optimization as line search for determine how far to move along a given direction.
Iterative algorithm

Most optimization problems cannot be solved in a closed form (a single step). For them, we develop iterative algorithms:

- start from an initial candidate solution: \( x^{(0)} \)
- generate a sequence of candidate solutions (iterates): \( x^{(1)}, x^{(2)}, \ldots, \)
- stop when a certain condition is met; return the candidate solution

In a large number of algorithms, \( x^{(k+1)} \) is generated from \( x^{(k)} \), that is, using the information of \( f \) at \( x^{(k)} \).

In some algorithms, \( x^{(k+1)} \) is generated from \( x^{(k)}, x^{(k-1)}, \ldots \). But, for time and memory consideration, most history iterates are not kept in memory.
Golden section search

**Problem:** given a closed interval $[a_0, b_0]$, a *unimodal* function (that is, having one and only one local minimizer in $[a_0, b_0]$), and only objective value information, find a point that is no more than $\epsilon$ away from that local minimizer.

Why make the unimodal assumption? Just for ease of illustration.
Golden section search

- **Mid-point:** evaluate the mid-point, that is, compute \( f\left(\frac{a_0+b_0}{2}\right) \). But, it cannot determine which half contains the local minimizer. **Does not work.**

- **Two-point:** evaluate at \( a_1, b_1 \in (a_0, b_0) \), where \( a_1 < b_1 \).
  1. if \( f(a_1) < f(b_1) \), then \( x^* \in [a_0, b_1] \);
  2. if \( f(a_1) > f(b_1) \), then \( x^* \in [a_1, b_0] \);
  3. if \( f(a_1) = f(b_1) \), then \( x^* \in [a_1, b_1] \). This case will rarely occur, so we can include it in either case 1 or case 2.
How to choose intermediate points

- **Symmetry:** length(left piece) = length(right piece), that is,
  
  \[(a_1 - a_0) = (b_0 - b_1)\]

- **Consistency:** Let \(\rho := \frac{a_1 - a_0}{b_0 - a_0} < \frac{1}{2}\). Such a ratio is maintained in subsequent iterations.

- **Reusing evaluated points:** the 1st iteration evaluates \(a_1, b_1\)
  
  - if the 1st iteration narrows to \([a_0, b_1]\), one of the next two points \(b_2\) shall equal \(a_1\)
    
    \[\implies \rho = \frac{b_1 - b_2}{b_1 - a_0} = \frac{b_1 - a_1}{b_1 - a_0}\]
  
  - if the 1st iteration narrows to \([a_1, b_0]\), one of the next two points \(a_2\) shall equal \(b_1\)
    
    \[\implies \rho = \frac{a_2 - a_1}{b_0 - a_1} = \frac{b_1 - a_1}{b_0 - a_1}\]
Putting together, using $b_1 - a_0 = 1 - \rho$, $b_1 - a_1 = 1 - 2\rho$, and $\rho < \frac{1}{2}$, we get

$$\rho(1 - \rho) = 1 - 2\rho \implies \rho = \frac{3 - \sqrt{5}}{2} \approx 0.382$$

Interestingly, observe the factor

$$\frac{1 - \rho}{1} = \frac{\rho}{1 - \rho} = \frac{\sqrt{5} - 1}{2} \approx 0.618,$$

which is the golden ratio by ancient Greeks. (Two segments with the ratio of the longer to the sum equals the ratio of the shorter to the longer.)
Example of golden section search

\[
\text{minimize } f(x) = x^4 - 14x^3 + 60x^2 - 70x.
\]

(one often omits the constant in the objective unless it helps simplify it)

- Initial interval: [0, 2].
- Required accuracy \( \epsilon = 0.3 \).
- 1st iteration: narrow to \([a_0, b_1] = [0, 1.236]\)
  
  ......

- 4th iteration: narrow to \([a_4, b_3] = [0.6525, 0.9443]\)
- STOP since \( b_3 - a_4 = 0.2918 < \epsilon \).
- No coincidence: \( b_3 - a_4 = 2(1 - \rho)^4 \). We can predict the number of iterations.
\[ x^4 - 14x^3 + 60x^2 - 70x \]

\[ x^* \approx 0.7809 \]
Algorithm complexity: How many iterations are needed?

Starting with an interval of length $L$, to reach an interval with length $\leq \epsilon$, we need $N$ iterations, where $N$ is the first integer such that

$$L(1 - \rho)^N \leq \epsilon \implies N \log(1 - \rho) \leq \log \frac{\epsilon}{L} \implies N \log \frac{1}{1 - \rho} \geq \log \frac{L}{\epsilon}.$$ 

Therefore,

$$N = \left\lceil C \log \frac{L}{\epsilon} \right\rceil,$$

where $C = \left( \log \frac{1}{1 - \rho} \right)^{-1}$.

We can write $N = O(\log \frac{1}{\epsilon})$ to emphasize its logarithmic dependence on $\frac{1}{\epsilon}$. 
Bisection method

**Problem:** given an interval \([a_0, b_0]\), a continuously differentiable, unimodal function, and derivative information, find a point that is no more than \(\epsilon\) away from the local minimizer.

**Mid-point works with derivative!**

Let \(x^{(0)} = \frac{1}{2}(a_0 + b_0)\)

- If \(f'(x^{(0)}) = 0\), then \(x^{(0)}\) is the local minimizer;
- If \(f'(x^{(0)}) > 0\), then narrow to \([a_0, x^{(0)}]\);
- If \(f'(x^{(0)}) < 0\), then narrow to \((x^{(0)}, b_0]\).

Every evaluation of \(f'\) reduces the interval by half. The reduction factor is \(\frac{1}{2}\). Need \(N\) iterations such that \(L(1/2)^N \leq \epsilon\), where \(L\) is the initial interval size and \(\epsilon\) is the targeted accuracy.

The *previous example* only needs 3 bisection iterations.
Newton’s method

Problem: given a twice continuously differentiable function and objective, derivative, and 2nd derivative information, find an approximate minimizer.

Newton’s method does not need intervals but must start sufficiently close to $x^*$

Iteration: minimize the quadratic approximation

$x^{(k+1)} \leftarrow \underset{x}{\text{arg min}} q(x) := f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2$.

(“arg min” returns the minimizer; “min” returns the minimal value.)

This iteration in a closed form

$$x^{(k+1)} := x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$$
Quadratic approximation with $f''(x) > 0$

$f$ and $q$ have the same value, tangent, and curvature at $x^{(k)}$
Example

The previous example

\[ f(x) = x^4 - 14x^3 + 60x^2 - 70x. \]

- True solution \( x^* \approx 0.7809 \)
- Initial point: \( x^{(0)} = 0 \).
  - \( x^{(1)} = 0.5833 \)
  - \( x^{(2)} = 0.7631 \)
  - \( x^{(3)} = 0.7807 \)
  - \( x^{(4)} = 0.7809 \)

Can produce highly accurate solutions in just a few steps.

Need just \( N = O(\log \log \left( \frac{1}{\epsilon} \right)) \) if \( f''(x^*) > 0 \) and \( x^{(0)} \) is sufficiently close.
What if \( f''(x) < 0 \)?

\[ f''(x) < 0 \text{ causes an ascending step} \]

In general, Newton’s iteration may diverge.
Newton’s method for finding zero

If we set \( g(x) := f'(x) \) and thus \( g'(x) = f''(x) \), then Newton’s method

\[
x^{(k+1)} := x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}
\]

is a way to solve \( g(x) = 0 \), that is, finding the zero of \( g \). Instead of a quadratic approximation to \( f \), the iteration uses a linear (tangent) approximation to \( g \).
A failed example

If \( x^{(0)} \) is not sufficiently close to \( x^* \), Newton’s sequence diverges unboundedly.

Note: \( g''(x) \) change signs between \( x^* \) and \( x^{(0)} \).
Secant method

Recall Newton’s method for minimizing \( f \) uses \( f''(x^{(k)}) \):

\[
x^{(k+1)} := x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.
\]

If \( f'' \) is not available or is expensive to compute, we can approximate

\[
f''(x^{(k)}) \approx \frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.
\]

This gives the iteration of the \textit{secant method}:

\[
x^{(k+1)} := x^{(k)} - \frac{(x^{(k)} - x^{(k-1)}) f'(x^{(k)})}{f'(x^{(k)}) - f'(x^{(k-1))}} = \frac{x^{(k)} f'(x^{(k-1)}) - x^{(k-1)} f'(x^{(k)})}{f'(x^{(k)}) - f'(x^{(k-1))}}.
\]

Note: the method needs \textit{two initial points}. 

The secant method is slightly slower than Newton’s method but is cheaper.
Comparisons of different 1D search methods

Golden section search (and Fibonacci search):

- one **objective** evaluation at each iteration,
- narrows search interval by less than half each time.

Bisection search:

- one **derivative** evaluation at each iteration,
- narrows search interval to exactly half each time.

Secant method:

- two points to start with; then one **derivative** evaluation at each iteration,
- must start near $x^*$, has **superlinear** convergence

Netwon’s method:

- one **derivative** and one **second derivative** evaluations at each iteration,
- must start near $x^*$, has **quadratic** convergence, fastest among the five
One-dimensional search methods are used as an important part in multi-dimensional optimization, often dubbed as the line search method.

At $x^{(k)}$, a method generates a search direction $d^{(k)}$.

- The gradient method sets $d^{(k)} = -\nabla f(x^{(k)})$.
- Newton’s method sets $d^{(k)} = -\left(\nabla^2 f(x^{(k)})\right)^{-1} \nabla f(x^{(k)})$.
- ... many other ways to set $d^{(k)}$ ...

Line search determines $\alpha_k$ that minimizes

$$\phi(\alpha) = f(x^{(k)} + \alpha d^{(k)})$$

and then sets

$$x^{(k+1)} := x^{(k)} + \alpha_k d^{(k)}.$$
At minimizer $\alpha_k$, $\nabla f(x^{(k)} + \alpha d^{(k)}) \perp d^{(k)}$
Practical line search

Line search is *inexact* in practice to save time.

Some practical acceptance rules for $\alpha_k$:

- **Armijo condition:**
  \[
  \phi_k(\alpha_k) \leq \phi_k(0) + c\alpha_k \phi_k'(0)
  \]
  where $c \in (0, 1)$ is often small, e.g., $c = 0.1$.

- **Armijo backtracking** starts from large $\alpha$ and decreases $\alpha$ geometrically until the Armijo condition is met.
• **Goldstein condition:**

\[ \phi_k(\alpha_k) \geq \phi_k(0) + \eta \alpha_k \phi'_k(0) \]

where \( \eta \in (c, 1) \), prevents tiny \( \alpha_k \).

• **Wolfe condition:**

\[ \phi'_k(\alpha_k) \geq \eta \phi'_k(0) \]

prevents \( \alpha_k \) from lending on a steep descending position.