LECTURE 18: ROOT-FINDING AND MINIMIZATION

STAT 545: Intro. to Computational Statistics

Vinayak Rao

Purdue University

November 22, 2016
Given some nonlinear function \( f : \mathbb{R} \rightarrow \mathbb{R} \), solve

\[
f(x) = 0
\]

Invariably need iterative methods.

Assume \( f \) is continuous (else things are really messy).

More we know about \( f \) (e.g. gradients), better we can do.

Better: faster (asymptotic) convergence.
Root bracketing

\[ f(a) \text{ and } f(b) \text{ have opposite signs } \rightarrow \text{ root lies in } (a, b). \]

\[ a \text{ and } b \text{ bracket the root.} \]

Finding an initial bracketing can be non-trivial. Typically, start with an initial interval and expand or contract.

Below, we assume we have an initial bracketing.
Root bracketing

\[ f(a) \text{ and } f(b) \text{ have opposite signs } \rightarrow \text{ root lies in } (a, b). \]

\[ a \text{ and } b \text{ bracket the root.} \]

Finding an initial bracketing can be non-trivial. Typically, start with an initial interval and expand or contract.

Below, we assume we have an initial bracketing.

Not always possible e.g. \[ f(x) = (x - a)^2 \] (in general, multiple roots/nearby roots lead to trouble).
Bisection method

Simplest root-finding algorithm.
Given an initial bracketing, cannot fail.
But is slower than other methods.

Successively halves the bracketing interval (binary search):

- Current interval = \((a, b)\)
- Set \(c = \frac{a+b}{2}\)
- New interval = \((a, c)\) or \((c, b)\)
  (whichever is a valid bracketing)
Bisection method

Simplest root-finding algorithm. Given an initial bracketing, cannot fail. But is slower than other methods.

Successively halves the bracketing interval (binary search):

- Current interval = \((a, b)\)
- Set \(c = \frac{a+b}{2}\)
- New interval = \((a, c)\) or \((c, b)\)
  (whichever is a valid bracketing)
Bisection method

Simplest root-finding algorithm. Given an initial bracketing, cannot fail. But is slower than other methods.

Successively halves the bracketing interval (binary search):

- Current interval = \((a, b)\)
- Set \(c = \frac{a+b}{2}\)
- New interval = \((a, c)\) or \((c, b)\) (whichever is a valid bracketing)
Bisection method

Simplest root-finding algorithm. Given an initial bracketing, cannot fail. But is slower than other methods.

Successively halves the bracketing interval (binary search):

- Current interval = \((a, b)\)
- Set \( c = \frac{a+b}{2} \)
- New interval = \((a, c)\) or \((c, b)\) (whichever is a valid bracketing)
Let $\epsilon_n$ be the interval length at iteration $n$. Upperbounds error in root.

$$\epsilon_{n+1} = 0.5 \epsilon_n \quad \text{(Linear convergence)}$$
Let $\epsilon_n$ be the interval length at iteration $n$. Upperbounds error in root.

$$\epsilon_{n+1} = 0.5 \epsilon_n$$ (Linear convergence)

Linear convergence:

- each iteration reduces error by one significant figure.
- every (fixed) $k$ iterations reduces error by one digit.
- error reduced exponentially with the number of iterations.
Let $\epsilon_n$ be the interval length at iteration $n$. Upperbounds error in root.

$$\epsilon_{n+1} = 0.5 \epsilon_n \quad \text{(Linear convergence)}$$

Linear convergence:

- each iteration reduces error by one significant figure.
- every (fixed) $k$ iterations reduces error by one digit.
- error reduced exponentially with the number of iterations.

Superlinear convergence:

$$\lim_{n \to \infty} |\epsilon_{n+1}| = C \times |\epsilon_n|^m \quad (m > 1)$$

Quadratic convergence:

Number of significant figures doubles every iteration.
Secant method and bisection method

Linearly approximate $f$ to find new approximation to root.

Secant method:
- Always keep the newest point
- Superlinear convergence ($\phi = \frac{1}{\sqrt{5}} \approx 0.618$, the golden ratio)
  $$\lim_{n \to \infty} \frac{\epsilon_{n+1}}{\epsilon_n} = \phi$$
- Bracketing (and thus convergence) not guaranteed.

False position:
- Can choose an old point that guarantees bracketing.
- Convergence analysis is harder.
Secant method and bisection method

Linearly approximate $f$ to find new approximation to root.

Secant method:

- always keep the newest point
- Superlinear convergence ($m = 1.618$, the golden ratio)

$$\lim_{n \to \infty} |\epsilon_{n+1}| = C \times |\epsilon_n|^{1.618}$$

- Bracketing (and thus convergence) not guaranteed.
Linearly approximate $f$ to find new approximation to root.

Secant method:

- always keep the newest point
- Superlinear convergence ($m = 1.618$, the golden ratio)
  \[
  \lim_{n \to \infty} |\epsilon_{n+1}| = C \times |\epsilon_n|^{1.618}
  \]
- Bracketing (and thus convergence) not guaranteed.
Secant method and bisection method

Linearly approximate $f$ to find new approximation to root.

Secant method:
- always keep the newest point
- Superlinear convergence ($m = 1.618$, the golden ratio)
  $$\lim_{n \to \infty} |\epsilon_{n+1}| = C \times |\epsilon_n|^{1.618}$$
- Bracketing (and thus convergence) not guaranteed.

False position:
- Can choose an old point that guarantees bracketing.
- Convergence analysis is harder.
In practice, people use more sophisticated algorithms.

Most popular is Brent’s method.

Maintains bracketing by combining bisection method with a quadratic approximation.

Lots of book-keeping.
At any point uses both function evaluation as well as derivative to form a linear approximation.

### Taylor expansion:

\[
f(x + h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \cdots
\]

Assume second- and higher-order terms are negligible.

Given \(x_i\), choose \(x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}\) so that \(f(x_{i+1}) = 0\):
NEWTON’S METHOD (A.K.A. NEWTON-RAPHSON)

At any point uses both function evaluation as well as derivative to form a linear approximation.

Taylor expansion: \( f(x + \delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \cdots \)
Newton’s method (a.k.a. Newton-Raphson)

At any point uses both function evaluation as well as derivative to form a linear approximation.

Taylor expansion: \[ f(x + \delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \cdots \]

Assume second- and higher-order terms are negligible.
Given \( x_i \), choose \( x_{i+1} = x_i + \delta \) so that \( f(x_{i+1}) = 0 \):
Newton’s method (a.k.a. Newton-Raphson)

At any point uses both function evaluation as well as derivative to form a linear approximation.

Taylor expansion: 
\[ f(x + \delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \cdots \]

Assume second- and higher-order terms are negligible.

Given \( x_i \), choose \( x_{i+1} = x_i + \delta \) so that \( f(x_{i+1}) = 0 \):

\[ 0 = f(x_i) + \delta f'(x_i) \]

\[ x_{i+1} = x_i - f(x_i)/f'(x_i) \]
Newton’s method (a.k.a. Newton-Raphson)

\[ x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \]
Newton’s method (a.k.a. Newton-Raphson)

\[ x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \]
Newton’s method (a.k.a. Newton-Raphson)

\[ x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \]
Letting $x$ be the root, we have

$$x_{i+1} - x = x_i - x - \frac{f(x_i)}{f'(x_i)}$$

$$\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)}$$
Letting $x$ be the root, we have

$$x_{i+1} - x = x_i - x - f(x_i)/f'(x_i)$$

$$\epsilon_{i+1} = \epsilon_i - f(x_i)/f'(x_i)$$

Also,

$$f(x_i) \approx f(x) + \epsilon_i f'(x) + \frac{\epsilon_i^2}{2} f''(x)$$
Convergence of Newton’s method

Letting $x$ be the root, we have

$$x_{i+1} - x = x_i - x - f(x_i)/f'(x_i)$$

$$\epsilon_{i+1} = \epsilon_i - f(x_i)/f'(x_i)$$

Also,

$$f(x_i) \approx f(x) + \epsilon_i f'(x) + \frac{\epsilon_i^2}{2} f''(x)$$

This gives

$$\epsilon_{i+1} = -\frac{f'(x_i)}{2f''(x_i)} \epsilon_i^2$$
Letting $x$ be the root, we have

$$x_{i+1} - x = x_i - x - \frac{f(x_i)}{f'(x_i)}$$

$$\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)}$$

Also,

$$f(x_i) \approx f(x) + \epsilon_i f'(x) + \frac{\epsilon_i^2}{2} f''(x)$$

This gives

$$\epsilon_{i+1} = -\frac{f'(x_i)}{2f''(x_i)} \epsilon_i^2$$

Quadratic convergence (assuming $f'(x)$ is non-zero at the root)
Away from the root the linear approximation can be bad. Can give crazy results (go off to infinity, cycles etc.) However, once we have a decent solution can be used to rapidly ‘polish the root’. Often used in combination with some bracketing method.
Find \((x_1, \cdots, x_N)\) such that:

\[
F_i(x_1, \cdots, x_N) = 0 \quad i = 1 \text{ to } N
\]

Much harder than the 1-d case.

Much harder than optimization.
Again, consider a Taylor expansion:

$$F(x + \delta x) = F(x) + J(x) \cdot \delta x + O(\delta x^2)$$

Here, $J(x)$ is the Jacobian matrix at $x$, with $J_{ij} = \frac{\partial F_i}{\partial x_j}$.
Again, consider a Taylor expansion:

\[ F(x + \delta x) = F(x) + J(x) \cdot \delta x + O(\delta x^2) \]

Here, \( J(x) \) is the Jacobian matrix at \( x \), with \( J_{ij} = \frac{\partial F_i}{\partial x_j} \).

Again, Newton’s method finds \( \delta x \) by solving \( F(x + \delta x) = 0 \)

\[ J(x) \cdot \delta x = -F(x) \]

Solve e.g. by LU decomposition.
Again, consider a Taylor expansion:

\[ F(x + \delta x) = F(x) + J(x) \cdot \delta x + O(\delta x^2) \]

Here, \( J(x) \) is the Jacobian matrix at \( x \), with \( J_{ij} = \frac{\partial F_i}{\partial x_j} \).

Again, Newton’s method finds \( \delta x \) by solving \( F(x + \delta x) = 0 \)

\[ J(x) \cdot \delta x = -F(x) \]

Solve e.g. by LU decomposition.

Iterate \( x_{\text{new}} = x_{\text{old}} + \delta x \) until convergence.

Can wildly careen through space if not careful.
Recall, we want to solve $\mathbf{F}(\mathbf{x}) = 0$ \quad ($F_i(\mathbf{x}) = 0, \quad i = 1 \cdots N$).
Recall, we want to solve $\mathbf{F}(\mathbf{x}) = 0 \quad (F_i(\mathbf{x}) = 0, \quad i = 1 \cdots N)$.

Minimize $f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{N} |F_i(\mathbf{x})|^2 = \frac{1}{2} |\mathbf{F}(\mathbf{x})|^2 = \frac{1}{2} \mathbf{F}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{x})$.

Note: It is NOT sufficient to find a local minimum of $f$. 
We move along $\delta x$ instead of $\nabla f = F(x)J(x)$.

This keeps our global objective in sight.
We move along $\delta x$ instead of $\nabla f = F(x)J(x)$.

This keeps our global objective in sight.

Note: $\nabla f \cdot \delta x = (F(x)J(x)) \cdot (-J^{-1}(x)F(x)) = -F(x)F(x) < 0$
A full Newton step sets $x_{new} = x_{old} + \delta x$.

This can cause $f$ to increase i.e. $f(x_{new}) > f(x_{old})$.

In this case, backtrack and set $x_{new} = x_{old} + \lambda \delta x$, $\lambda \in (0, 1)$.

Since $\delta x$ is a descent direction, there exists a sufficiently small $\lambda$ that causes $f$ to decrease.
A full Newton step sets $x_{new} = x_{old} + \delta x$.

This can cause $f$ to increase i.e. $f(x_{new}) > f(x_{old})$.

In this case, backtrack and set $x_{new} = x_{old} + \lambda \delta x$, $\lambda \in (0, 1)$.

Since $\delta x$ is a descent direction, there exists a sufficiently small $\lambda$ that causes $f$ to decrease.

Finding best $\lambda$: too much work usually.

However, just causing $f$ to decrease is not sufficient.
Wolfe conditions

Big steps with little decrease

Small steps getting us nowhere

\[ f(x) \]

\[ \nabla f(x) \]

\[ c_1, c_2 \in (0; 1) \]

\[ g(0; 1) \]

\[ f(x) \]

\[ \nabla f(x) \]

\[ c_2 \nabla f(x) \]

\[ \frac{1}{16} = \frac{1}{17} \]
Wolfe conditions

Big steps with little decrease

Small steps getting us nowhere

Avg. decrease at least some fraction of initial rate:

\[
f(x + \lambda \delta x) \leq f(x) + c_1 \lambda (\nabla f \cdot \delta x), \quad c_1 \in (0, 1) \text{ e.g. } 0.9
\]
Wolfe Conditions

Big steps with little decrease

Small steps getting us nowhere

Avg. decrease at least some fraction of initial rate:

\[ f(x + \lambda \delta x) \leq f(x) + c_1 \lambda (\nabla f \cdot \delta x), \quad c_1 \in (0, 1) \text{ e.g. } 0.9 \]

Final rate is greater than some fraction of initial rate:

\[ \nabla f(x + \lambda \delta x) \cdot \delta x \geq c_2 \nabla f(x) \delta x, \quad c_2 \in (0, 1) \text{ e.g. } 0.1 \]
Permissible $\lambda$’s under condition 1
Wolfe conditions

Permissible λ’s under condition 2

$\nabla f$

$c_2 \nabla f$