STAT 7650 - Computational Statistics Lecture Slides

Optimization and Solving Nonlinear Equations (Root Finding)

Elvan Ceyhan

Updated: February, 2025

AU

- Based on parts of: Chapter 2 in Givens & Hoeting (Computational Statistics), and Chapter 14 of Lange (Numerical Analysis for Statisticians).

Outline

Motivation

- In statistical applications, point estimation problems often boil down to maximizing a function:
  - Maximum likelihood
  - Least squares
  - Maximum a posteriori
- When the function to be optimized is "smooth," we can reformulate optimization into a root-finding problem.
- **Problem:** These problems often have *no analytical solution*.
- Therefore, we need *numerical tools* to solve them.

General Setup

- **Two kinds of problems:**
  - Root-finding: Solve $f(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathbb{R}^d, d \geq 1$.
  - Optimization: Maximize $f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d, d \geq 1$.
- Equivalent if you need to solve $f'(\mathbf{x}) = 0$.
- We will address univariate and multivariate cases separately.
- Methods construct a sequence $\{\mathbf{x}_t : t = 0, 1, 2, \ldots\}$ designed to converge (as $t \to \infty$) to the solution, denoted by $\mathbf{x}^*$.

General Setup (cont'd)

Theoretical considerations:

- Under what conditions on f' (or f) and initial guess $\mathbf{x}_0$ can we prove that $\mathbf{x}_t \to \mathbf{x}^*$?
- If $\mathbf{x}_t \to \mathbf{x}^*$, then how fast is the convergence, i.e., what is its convergence order?

Practical considerations:

- How to write and implement the algorithm?
- Can't run the algorithm till $t = \infty$, so when to stop?

Convergence Criteria

- The *convergence criteria* is usually something like:

$$|x_{\text{new}} - x_{\text{old}}| < \varepsilon \quad \text{i.e.} \quad |x_{t+1} - x_t| < \varepsilon$$

  where ε is a specified small number, e.g., $\varepsilon = 10^{-7}$.

- A *relative convergence criteria* might be better:

$$\frac{|x_{\text{new}} - x_{\text{old}}|}{|x_{\text{old}}|} < \varepsilon$$

Relative Convergence in Optimization

Definition: Relative convergence refers to stopping conditions that consider the **relative change** in function values or parameter updates rather than absolute changes.

- Useful when function values or parameters have large or varying magnitudes.
- Ensures stopping criteria are scale-invariant.

Common Relative Convergence Criteria

1. Relative Change in Objective Function

$$\frac{|f(x_k) - f(x_{k-1})|}{|f(x_{k-1})|} < \delta$$

Ensures the function value is stabilizing in proportion to its magnitude.

Common Relative Convergence Criteria (cont.)

2. Relative Change in Variables

$$\frac{\|x_k - x_{k-1}\|}{\|x_{k-1}\|} < \eta$$

Useful when variables vary significantly in scale.

Common Relative Convergence Criteria (cont.)

3. Relative Gradient Norm

$$\frac{\|\nabla f(x_k)\|}{\|\nabla f(x_0)\|} < \epsilon$$

Ensures that the optimization process is making proportionate improvements.

Why Use Relative Convergence?

- Works well when function values or variables are large.
- Prevents premature stopping when dealing with different scales.
- Ensures that improvements are meaningful in proportion to their magnitude.

Definition of Order of Convergence

An algorithm has **order of convergence** β if:

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

where:

- $\epsilon(t)$ is the error at iteration t.
- $\beta > 0$ measures how **quickly** the error shrinks.
- $c \neq 0$ is a constant.

Higher β means faster convergence!

Connection to Convergence Rates in Optimization

The order of convergence relates to well-known convergence rates:

- $\beta = 1 \Rightarrow$ **Linear Convergence** (error shrinks proportionally)
- $1 < \beta < 2 \Rightarrow$ **Superlinear Convergence** (faster than linear)
- $\beta = 2 \Rightarrow$ **Quadratic Convergence** (error squared at each step)
- $0 < \beta < 1 \Rightarrow$ **Sublinear Convergence** (very slow)

Example: Newton's method is **quadratically convergent** under good conditions.

Why Use lim sup Definition?

The order of convergence is often written as:

$$\limsup_{t\to\infty} \frac{|\epsilon(t+1)|}{|\epsilon(t)|^{\beta}} \leq C$$

where $C > 0$ ensures the worst-case asymptotic behavior.

- Allows for **variability** in error reduction per iteration.
- Ensures **robustness** in practical optimization problems.
- Captures **asymptotic behavior** for sufficiently large t.

Why Use $\leq C$ Instead of $= C$?

Using $\leq C$ instead of $= C$ allows for:

- **Generalization**: Convergence behavior may not follow strict proportionality.
- **Flexibility**: Accounts for fluctuations in the error sequence.
- **Realism**: Many practical algorithms exhibit varying convergence rates.

This ensures that the convergence definition applies to more cases.

Outline

Univariate Optimization

Optimizing smooth univariate functions

- Bisection
- Newton's method
- Fisher scoring
- Secant method
- (scaled) Fixed point iteration

Goal: Maximize a real-valued function $f(x)$.

$f(x)$ may be a likelihood, a profile likelihood, a Bayesian posterior, or some other function (of interest).

Univariate Optimization for Smooth g

Example 1:

Maximize

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

with respect to x.

We cannot find the root of $f'(x) = \dfrac{1 + 1/x - \log x}{(1 + x)^2}$ analytically.

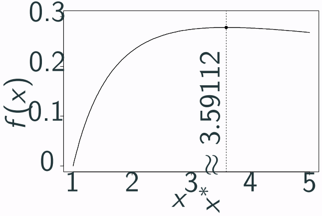

Figure 1: The maximum of $f(x) = \dfrac{\log x}{1 + x}$ occurs at $x^* \approx 3.59112$, indicated by the vertical line.

Example 2:

The following data are an i.i.d. sample from a Cauchy$(\theta, 1)$ distribution:

1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2.40, 4.53, -0.07, -1.05, -13.87, -2.53, -1.75, 0.27, 43.21.

The likelihood function is

$$\prod_{i=1}^{20} \frac{1}{\pi\left(1+(x_i-\theta)^2\right)}. \tag{2}$$

Find the MLE for θ.

The score function (first derivative of the log-likelihood) has multiple roots requiring numerical solution.

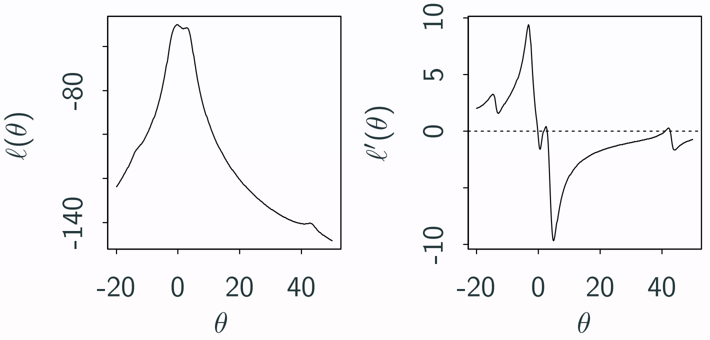

Figure 2: Log likelihood and score function for the Cauchy data.

Outline

Bisection - Basic Idea

- Find a root x^* of f in interval $[a, b]$.
  - **Claim:** If f is continuous on $[a, b]$ and $f(a)f(b) \leq 0$ then the *intermediate value theorem*, then there exists a root $x^* \in (a, b)$. Why?
- Pick an initial guess $x_0 = \frac{a+b}{2}$.
  - Then x^* must be in either $[a, x_0]$ or $[x_0, b]$.
  - Evaluate $f(x)$ at the endpoints to determine which one.
- The selected interval, call it $[a_1, b_1]$, is now just like the initial interval; i.e., we know it must contain x^*.
  - Take $x_1 = \frac{a_1+b_1}{2}$.
- Continue this process to construct a sequence $\{x_t : t = 0, 1, 2, \ldots\}$.

Bisection Algorithm

For the given $f(x)$, assume the interval at the t-th step is $[a_t, b_t]$ are given.

1. Set $x_t = \frac{a_t+b_t}{2}$.
2. If $f(a_t)f(x_t) \leq 0$, then $b_{t+1} = x_t$ and $a_{t+1} = a_t$; else $a_{t+1} = x_t$ and $b_{t+1} = b_t$.
3. If "converged," then stop; otherwise return to Step 1.

---

1. In *computer code*, you first initialize $a = a_0$ and $b = b_0$ and update as follows at each step
2. Set $x = \frac{a+b}{2}$.
3. If $f(a)f(x) \leq 0$, then $b = x$; else $a = x$.
4. If "converged," then stop; otherwise go to Step 1.

Note: There is also a related algorithm called Golden Section Search Algorithm.

Bisection Theory

- **Claim:** If f is continuous, then $x_t \to x^*$.
  - **Proof:**
    - If $[a_t, b_t]$ is the bounding interval at step t, then $f(a_t)f(b_t) \leq 0$ and $\lim_{t\to\infty} a_t = \lim_{t\to\infty} b_t$.
    - So, x_t converges to some $\tilde{x}$, and by continuity $f(\tilde{x})^2 \leq 0$.
    - Then $f(\tilde{x}) = 0$ and, since x^* is the unique root, $\tilde{x} = x^*$. $\square$
- Convergence holds under very mild conditions of f, but the robustness comes at the price of the order of convergence.

Examples

- Find x^* to maximize the function

$$f(x) = \frac{\log x}{1+x}, \quad x \in [1,5].$$

  - Note that

$$f'(x) = \frac{1 + x^{-1} - \log x}{(1+x)^2}.$$

- Find the $100p$-th percentile of a Student-t distribution, i.e.,
  - find x^* such that $F(x^*) = p$, where F is the t-distribution function, with degrees of freedom $df = \nu$ fixed.

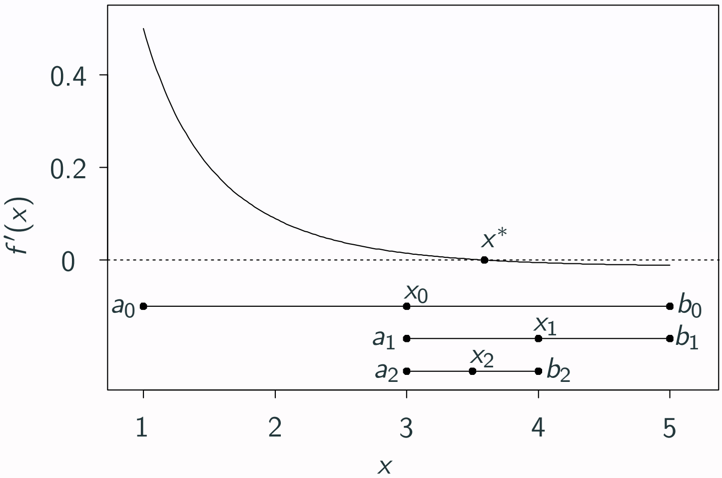

Figure 3: Illustration of the bisection method. The top portion of this graph shows $f'(x)$ and its root at x^*. The bottom portion shows the first three intervals obtained using the bisection method with $[a_0, b_0] = [1,\ 5]$. The t-th estimate of the root is at the center of the t-th interval.

Outline

Basic Idea

- **Newton's Method** is usually presented in a calculus class.
- Idea is *to approximate a nonlinear function near its root by a linear function* which can be solved by hand.
  - Recall that Taylor's Theorem gives the linear approximation of a function $f(x)$ in a neighborhood of some point x_0 as

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0).$$

  - Setting this approximation equal to 0 and solving gives

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Newton Method - Algorithm

- Assume the function $f(x)$, its derivative $f'(x)$, and an initial guess x_0 are given. Set $t = 0$.
  1. At step t (so, we have x_t already computed), set
  $$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}.$$
  2. If the convergence criteria is met, then stop; otherwise, set $t = t + 1$ and return to Step 1.
- **Caveats:**
  - Convergence depends on the choice of x_0 and shape of f.
  - Unlike bisection, Newton's Method might not converge!

Newton Method - Theory

- **Claim:** If f'' is continuous and x^* is a root of f, with $f'(x^*) \neq 0$, then there exists a neighborhood N of x^* such that Newton's Method converges to x^* for any $x_0 \in N$.
  - Proof uses Taylor approximation.
  - Proof also shows that the convergence order is quadratic.
- Other results about Newton's Method are available; see HW.
- If Newton Method converges, then it's faster than bisection, but added speed has a cost:
  - Requires differentiability and the derivative f'.
  - Convergence is sensitive to the choice of x_0.

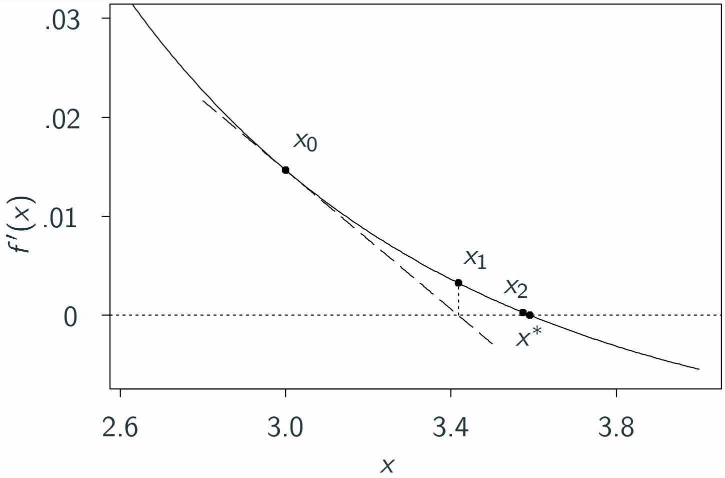

Figure 4: Illustration of Newton's Method applied to maximize the function in Equation (1). At the first step, Newton's method approximates f' by its tangent line at x_0 whose root, x_1, serves as the next approximation of the true root, x^*. The next step similarly yields x_2, which is already quite close to the root at x^*.

Speed is not the only factor to consider.

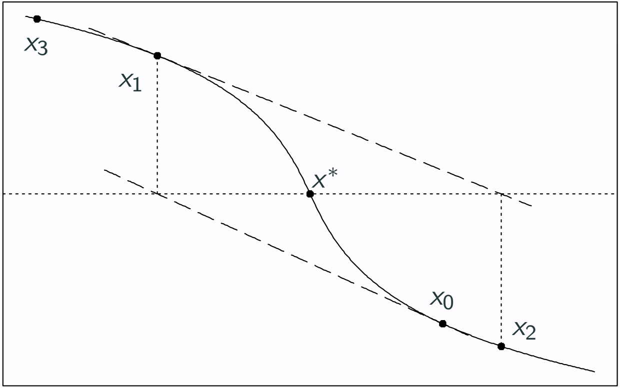

Figure 5: Starting from x_0, Newton's method diverges by taking steps that are increasingly distant from the true root, x^*.

Bisection would have found this root easily.

Starting values are also critical.

Cauchy Example

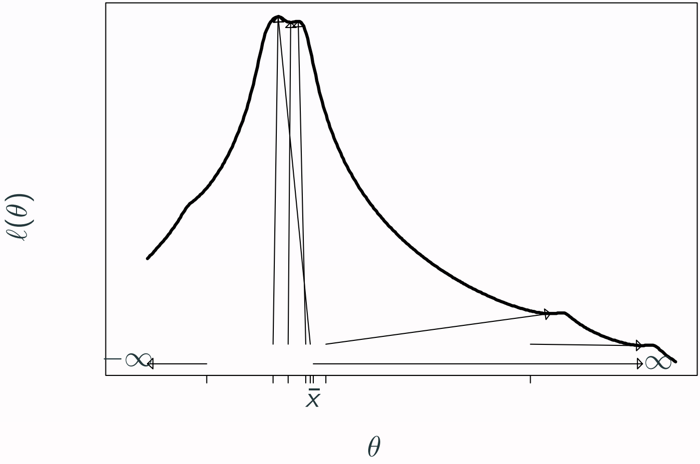

Figure 6: Log-likelihood for the Cauchy data. Arrows show convergence of Newton's method from several starting values.

Outline

Fisher Scoring

- In maximum likelihood applications, the goal is to find roots of the log-likelihood function, i.e., $\ell'(\hat{\theta}) = 0$.
- In this context, Newton's Method looks like

$$\theta_{t+1} = \theta_t - \frac{\ell'(\theta_t)}{\ell''(\theta_t)}, \quad t = 0, 1, 2, \ldots.$$

- But recall that $-\ell''(\theta)$ is an approximation to the Fisher information $I_n(\theta)$.
- So, can rewrite Newton's Method as

$$\theta_{t+1} = \theta_t + \frac{\ell'(\theta_t)}{I_n(\theta_t)}, \quad t = 0, 1, 2, \ldots.$$

- This modification is called *Fisher Scoring*.

Outline

Secant Method - Basic Idea

- Newton's Method requires a formula for $f'(x)$.
- To avoid this, approximate $f'(x)$ at x_0 by a difference ratio.
  - That is, recall from calculus that

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad \text{for } h \text{ small and positive.}$$

- Then the *secant method* follows Newton's Method exactly, except we substitute a difference ratio for $f'(x)$.
- Name is because *the linear approximation is a secant*, not a tangent line.

Secant Method - Algorithm

- Suppose $f(x)$ and two initial guesses x_0, x_1 are given. Set $t = 1$.
  1. At step t, calculate
  $$x_{t+1} = x_t - \frac{f(x_t)}{\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}}}.$$
  2. If the convergence criteria are satisfied, then stop; else, set $t = t + 1$ and return to Step 1.
- Two initial guesses are needed because the difference ratio in the first iteration requires two values.
- Can be unstable at early iterations because the *difference ratio* may be a poor approximation of f'; reasonable sacrifice if f' is not available.
- If the secant method converges, the order is almost quadratic.

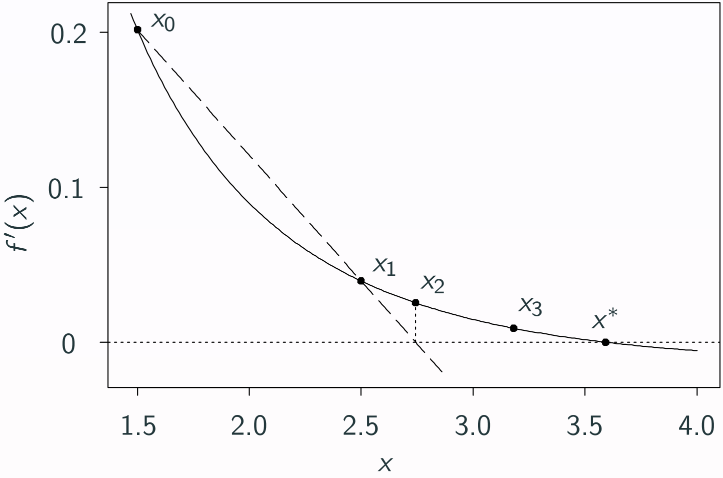

Figure 7: The secant method locally approximates f' using the secant line between x_0 and x_1. The corresponding estimated root, x_2, is used with x_1 to generate the next approximation.

Outline

Fixed-Point Iteration - Basic Idea

- Some problems require finding a fixed-point, i.e., a point x^* such that $F(x^*) = x^*$.
- A root-finding problem can be written as a fixed-point problem with $F(x) = f(x) + x$.
- The function $F(x)$ is a contraction, if,
  - $F(x) \in [a, b]$ for all $x \in [a, b]$
  - $$|F(x) - F(y)| \leq \alpha |x - y|, \quad \text{for } 0 < \alpha < 1 \text{ for all } x, y \in [a, b]$$

  then the point $F(x)$ will be closer to $x^* = F(x^*)$ than x.
- Banach's Fixed-Point Theorem says:
  - Contraction mappings have a unique fixed point x^*, and
  - From a starting point x_0, the iterates $x_{t+1} = F(x_t)$ will converge to x^*.

Fixed-Point Iteration - Algorithm

- Suppose $F(x)$ and an initial guess x_0 are given. Set $t = 0$.
    1. Calculate $x_{t+1} = F(x_t)$.
    2. If convergence criterion is met, then stop; else, set $t = t + 1$ and return to Step 1.
- It can be shown that

$$|F(x_t) - x^*| \leq \alpha^t |x_0 - x^*|,$$

  so, fixed-point iteration *converges at a geometric rate*.
- If using fixed-point methods for root-finding, $F(x) = f(x) + x$ may not be the best choice; for example, maybe a scaled version would be better.

Example - Kepler's Equation

- Kepler's Equation in orbital mechanics says

$$x = M + \varepsilon \sin x,$$

where M and $\varepsilon \in (0, 1)$ are fixed quantities.[1]

- Our goal is to solve for x, given M and ε.
  - Write $F(x) = M + \varepsilon \sin x$.
  - Then $F'(x) = \varepsilon \cos x$ and $|F'(x)|$ is uniformly bounded by ε.
- So, F is a contraction and fixed-point iteration will converge to a solution to Kepler's equation.

[1] See Wikipedia page on Kepler's equation for more info.

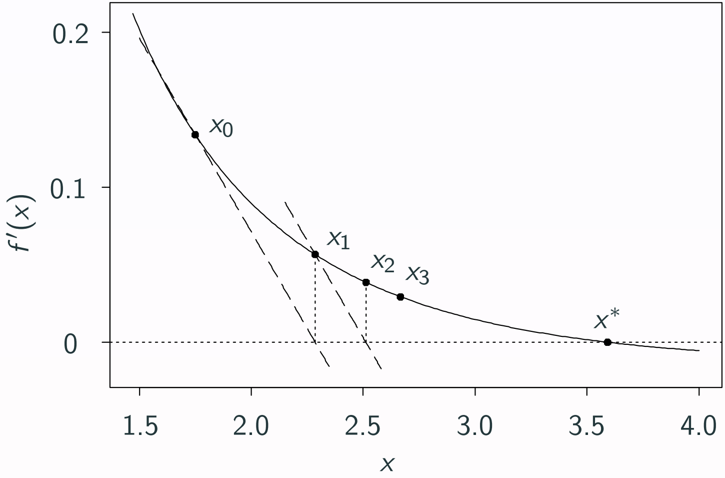

Figure 8: The first three steps of scaled fixed point iteration to maximize $f(x) = \dfrac{\log x}{1+x}$ using $F(x) = c\, f'(x) + x$ with scale parameter $c = 4$.

Outline

Root-Finding and Optimization in R

- **Univariate Problems:**
  - `uniroot` does root-finding.
  - `optimize` does optimization.
  - See documentation files (and the `R` code on Canvas) for details.
- **Multivariate Problems:**
  - `nlm` does non-linear **min**imization with Newton-like methods.
  - `optim` is maybe a better choice.
- More on these later.

A Note About Constraints

- The univariate methods built into `R` are not particularly good at handling optimization problems where the parameter x is constrained, e.g., if x must be non-negative.
- The built-in `R` routines assume x has no constraints, so to be safe you may want to write your functions this way.
- For example, if x is required to be non-negative, then reparametrize as $y = \log x$, and set $g(y) = f(e^y)$ and perform the optimization on the function $g(y)$. If y^* is the optimizer value, then $x^* = e^{y^*}$ will be the optimizer in the original optimization problem.
- Don't forget: Re-parametrization will affect derivatives!

Outline

Multivariate Optimization

Optimizing smooth multivariate functions

- Newton's method
- Fisher scoring
- ascent algorithms
- discrete Newton method
- (scaled) fixed point iteration
- quasi-Newton methods
- Gauss-Newton method
- nonlinear Gauss-Seidel iteration
- Nelder–Mead algorithm

Multivariate Optimization

- In the univariate optimization part, we posed the problem as root finding problem for a function f, which was an optimization problem when $f = g'$ (where g is the function to maximize).
- In the multivariate optimization part, we will directly pose the problem as root finding problem for a function f', but notice that we are still solving the problem of root finding:r
  - $\max_{\mathbf{x}} f(\mathbf{x})$ over $\mathbf{x}$
    is equivalent to
    finding the root of $f'(\mathbf{x})$.

Outline

Newton's Method - More Than One Variable

- Suppose now that $f(\mathbf{x})$ is a function of several variables, say $\mathbf{x} = (x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$.
- Newton's Method works exactly the same as before, just the derivatives are more complicated.
  - $f'(\mathbf{x})$ is the gradient — vector of first partial derivatives.
  - $f''(\mathbf{x})$ is the *Hessian* — matrix of second partial derivatives.
- Based on (MV) Taylor's Formula again, Newton's Method is

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

- If we are maximizing a log-likelihood, $\ell(\boldsymbol{\theta})$, then the Fisher Scoring adjustment is just like before.

Example: Gamma Distribution MLE

- $X_1, \ldots, X_n \overset{iid}{\sim} \text{Gamma}(\alpha, \beta)$, where both α and β are unknown parameters to be estimated.
- Density function is

$$f(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x \geq 0.$$

- The log-likelihood function is (effectively)

$$\ell(\alpha, \beta) = n\alpha \log \beta - n \log \Gamma(\alpha) + \alpha \sum_{i=1}^{n} \log X_i - \beta \sum_{i=1}^{n} X_i.$$

- Find first and second (partial) derivatives of $\ell(\alpha, \beta)$.
- `R` code on Canvas implements Newton's Method to find the MLE.

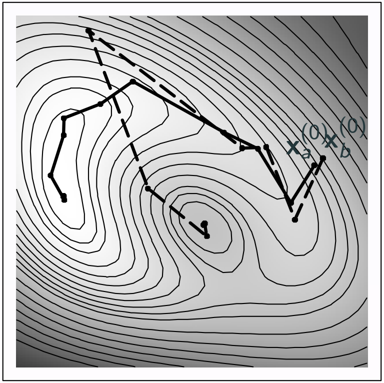

Figure 9: An application of Newton's method for maximizing a complex bivariate function. The surface of the function is indicated by shading and contours, with light shading corresponding to high values. Two runs starting from $\mathbf{x}_a^{(0)}$ and $\mathbf{x}_b^{(0)}$ are shown. These converge to 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.

Outline

Motivation for Alternatives

- Newton's Method is a very good technique for both univariate and multivariate optimization.
  - Difficulty in the multivariate case is the derivation and/or computation of the Hessian matrix and its inverse.
- Is it possible to use some other matrix, say $M^{(t)}$, in place of the Hessian $f''(\mathbf{x}^{(t)})$?
  - Yes, and we will discuss a few such methods:
    - Ascent methods
    - Discrete Newton and fixed-point methods
    - Quasi-Newton methods

Ascent Methods

- Fix matrices $M^{(t)}$ and numbers $\alpha^{(t)}$, $t = 0, 1, 2, \ldots$.
- Ascent methods look like

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha^{(t)}[M^{(t)}]^{-1} f'(\mathbf{x}^{(t)}).$$

- Goal is to choose $M^{(t)}$ and $\alpha^{(t)}$ such that the function increases when $\mathbf{x}^{(t)}$ is updated to $\mathbf{x}^{(t+1)}$.
- It follows from Taylor's Formula that, if $-M^{(t)}$ is positive definite and $\alpha^{(t)}$ is sufficiently small, then ascent occurs.

Ascent Methods (cont'd)

- Method of *steepest ascent* takes $M^{(t)} \approx -I_p$.
- Motivation is the basic fact from multivariable calculus that the *gradient points in the direction of steepest ascent*.
- Then the updating equation looks like

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha^{(t)} f'(\mathbf{x}^{(t)}), \quad t = 0, 1, 2, \ldots.$$

- How to pick a good $\alpha^{(t)}$?
  - A *backtracking* approach determines $\alpha^{(t)}$ iteratively:
    1. Start with $\alpha^{(t)} = 1$.
    2. Update $\mathbf{x}^{(t+1)}$ with this $\alpha^{(t)}$.
    3. If ascent occurs, then increment t; otherwise, set $\alpha^{(t)} = \alpha^{(t)}/2$ and go back to Step 2.

Ascent Methods (cont'd)

- **Claim:** If $\alpha^{(t)}$ is sufficiently small, then ascent occurs.
- **Sketch of Proof:**
  - From the two-term Taylor expansion of $f(\mathbf{x}^{(t+1)})$ near $\mathbf{x}^{(t)}$, we have

$$f(\mathbf{x}^{(t+1)}) = f(\mathbf{x}^{(t)}) + f'(\mathbf{x}^{(t)})^T(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}) + \frac{1}{2}(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})^T f''(\tilde{\mathbf{x}})(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})$$

    where $\tilde{\mathbf{x}}$ is between $\mathbf{x}^{(t+1)}$ and $\mathbf{x}^{(t)}$.
  - Plug in definition of $\mathbf{x}^{(t+1)}$; then $f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})$ is

$$\alpha^{(t)}\|f'(\mathbf{x}^{(t)})\|^2 + \frac{1}{2}(\alpha^{(t)})^2 f'(\mathbf{x}^{(t)})^T f''(\tilde{\mathbf{x}}) f'(\mathbf{x}^{(t)})$$

  - Second term is $\approx c(\alpha^{(t)})^2\|f'(\mathbf{x}^{(t)})\|^2$, where $c \in \mathbb{R}$.
  - Make $\alpha^{(t)}$ small enough that bound is positive. $\square$

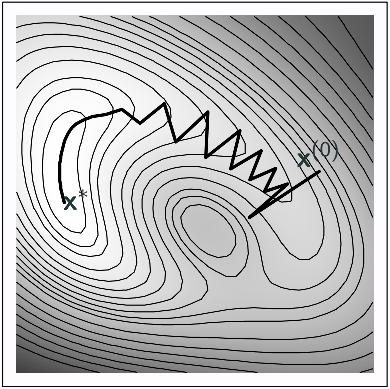

Figure 10: Steepest ascent with backtracking, using $\alpha = 0.25$ initially at each step.

The ascent direction is not necessarily the wisest, and backtracking doesn't prevent oversteps.

Discrete Newton and Fixed-Point Methods

- If we use an initial approximation, we get a MV fixed-point method.
- For example, with a fixed matrix M, write

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

  - A reasonable choice is $M = f''(\mathbf{x}_0)$.
- Replace Hessian $f''(\mathbf{x})$ in Newton with a discrete approximation (using difference ratios) gives a discrete Newton method.
  - Can be expensive — each step requires lots of difference ratios.

Quasi-Newton Methods

- Recall that the general idea is to replace the Hessian with some reasonable approximation.
- Methods so far have not made a serious attempt to capture any real information about f in the matrix $M^{(t)}$.
- How to ensure that $M^{(t)}$ somehow approximates the Hessian?
  - A secant condition can do the job:

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

  - How to construct a matrix sequence $M^{(t)}$ that satisfies this?

Quasi-Newton Methods (cont'd)

- There are classes of matrices that satisfy the secant condition.
  - There is a unique symmetric rank-one update:

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

    where

    $$\begin{aligned}
    \mathbf{v}^{(t)} &= \mathbf{y}^{(t)} - M^{(t)}\mathbf{z}^{(t)}, \\
    \mathbf{z}^{(t)} &= \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}, \\
    \mathbf{y}^{(t)} &= f'(\mathbf{x}^{(t+1)}) - f'(\mathbf{x}^{(t)}), \\
    c^{(t)} &= \frac{1}{(\mathbf{v}^{(t)})^T\mathbf{z}^{(t)}}.
    \end{aligned}$$

- The go-to approach is a rank-two update, called *BFGS*.
  - Formula is messy — see Equation (2.50) in the textbook.
- The `R` code on Canvas implements BFGS; more on `R` below.

Example: Problem 2.3 in G&H

- Survival analysis problem, with censored data.
- Data (y_i, x_i, w_i) where
  - y_i is the recorded survival time,
  - x_i is a treatment versus control indicator,
  - w_i is a real versus censored survival time indicator.
- Proportional hazards model gives log-likelihood

$$\ell(\theta) = \sum_{i=1}^{n} \left[w_i \log(\lambda_i) - \lambda_i + w_i \log\left(\frac{y_i}{\lambda_i}\right) \right],$$

  where $\lambda_i = y_i e^{\beta_0 + \beta_1 x_i}$.
- **Goal:** Find MLE of $\boldsymbol{\theta} = (\beta_0, \beta_1)$.

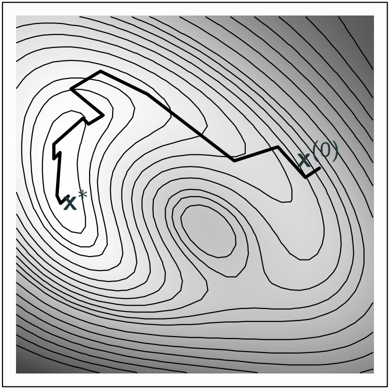

Figure 11: Quasi-Newton optimization with the BFGS update and backtracking to ensure ascent.

Convergence of quasi-Newton methods is generally superlinear, but not quadratic. These are powerful and popular methods, used, for example, see `nlmin()` *in R.*

Outline

Least Squares

- Suppose that the function to maximize is quadratic, e.g.,

$$f(\mathbf{x}) = -\|\mathbf{y} - A\mathbf{x}\|^2.$$

- We can solve this one analytically:

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{y}.$$

- This is the least squares solution you may have seen in a linear algebra or numerical analysis course.

Gauss-Newton for Least Squares

- In the previous slide, the goal basically was to approximate **y** by a linear function of **x**.
- What if the function is non-linear?
- **Gauss-Newton Method:**
  - Consider $g(\theta) = -\sum_{i=1}^{n}\{y_i - f(\mathbf{x}_i, \theta)\}^2$.
  - Fix θ_0 and approximate $\theta \mapsto f(\mathbf{x}_i, \theta)$ (i.e., $h(\theta) = f(\mathbf{x}_i, \theta)$) by a linear function, that is,

$$f(\mathbf{x}_i, \theta) = f(\mathbf{x}_i, \theta_0) + f'(\mathbf{x}_i, \theta_0)(\theta - \theta_0).$$

  - Plug this in for $f(\mathbf{x}_i, \theta)$ in $g(\theta)$ and note the similarity to the least squares problem.
  - Solve analytically for θ; call the solution θ_1 and redo.

Gauss-Newton Method (alternate take)

- Address nonlinear least squares problems for observed data $(y_i, \mathbf{x}_i)$ with model $Y_i = f(\mathbf{x}_i, \boldsymbol{\theta}) + \epsilon_i$.
- **Objective:** Maximize $g(\boldsymbol{\theta}) = -\sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \boldsymbol{\theta}))^2$.
- Newton's method approximates g via Taylor series. But Gauss-Newton approximates f by its linear Taylor expansion about $\boldsymbol{\theta}^{(t)}$, leading to $Y_i = \tilde{f}(\mathbf{x}_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}) + \tilde{\epsilon}_i$.

where

$$\tilde{f}(\mathbf{x}_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}) = f(\mathbf{x}_i, \boldsymbol{\theta}^{(t)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^T \mathbf{f}'(\mathbf{x}_i, \boldsymbol{\theta}^{(t)})$$

with for each i, $\mathbf{f}'(\mathbf{x}_i, \boldsymbol{\theta}^{(t)})$ is the column vector of partial derivatives of f with respect to $\theta_j^{(t)}$, for $j = 1, \ldots, p$, evaluated at $(\mathbf{x}_i, \boldsymbol{\theta}^{(t)})$.

Gauss-Newton Method (cont'd)

- Maximize approximated objective
$$\tilde{g}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left(y_i - \tilde{f}(\mathbf{x}_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}) \right)^2.$$
- $Y_i = \tilde{f}(\mathbf{x}_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}) + \tilde{\epsilon}_i$ can be written as follows
$$\mathbf{X}^{(t)} = \mathbf{A}^{(t)}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) + \tilde{\boldsymbol{\epsilon}}$$
where $x_i^{(t)} = y_i - f(\mathbf{x}_i, \boldsymbol{\theta}^{(t)})$ is the working response, and $\mathbf{a}_i^{(t)} = \mathbf{f}'(\mathbf{x}_i, \boldsymbol{\theta}^{(t)})$ is the i-th row of $\mathbf{A}^{(t)}$.
- **This is a regression problem!** Thus, the update rule is:
$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \left((\mathbf{A}^{(t)})^T \mathbf{A}^{(t)} \right)^{-1} (\mathbf{A}^{(t)})^T \mathbf{x}^{(t)}.$$
- Efficient, no Hessian computation needed, best for fairly well-fitting, not severely nonlinear models.

Outline

Built-in Functions in R

- `R` has two built-in functions for optimization:
  - `nlm` for non-linear minimization.
  - `optim` for optimization.
- Functions in `R` are designed to do minimization.
- I don't use `nlm` much, mostly `optim` with method='BFGS'.
- See the `R` code on Canvas, and also documentation on `optim`.

Outline

Root-Finding with Noise

- The tools described above all require that the function can be evaluated exactly.
- However, there are some problems where there is some error in evaluating the function, e.g., maybe we can only get a Monte Carlo approximation of the function.
- In such cases, Newton-like methods cannot be used.
- A neat generalization of Newton Methods to handle noisy functions is called *stochastic approximation*.
- We may discuss this briefly in the Monte Carlo Section.

Non-Differentiable Functions

- The methods described above all are based on the assumption that the function $f(\mathbf{x})$ to be optimized has at least one derivative.
- But there are problems where this assumption does not hold:
    - Quantile regression.
    - Regularized regression with, say, the *lasso*.
- For these problems, different tools are needed, e.g.,
    - Linear programming.
    - Iterative re-weighted least squares.

Functions on Discrete Spaces

- Non-differentiability is one thing, but what if the function is only defined on a discrete space?
  - In this case, the derivative doesn't even make sense.
- If there are only a few possible **x** values then, of course, it's easy to find the maximum.
- But what if there are billions of points? It's not unreasonable to have problems with $2^{50} \approx 10^{14}$ points. In such cases, it's impossible to search them all!
- These are called *combinatorial optimization* problems, and one interesting algorithm is called *simulated annealing*.
- Chapter 3 in the textbook discusses these issues.