MA 6973 Computational Statistics Sep 15, 2021

Gentle, Ch. 6

Basic NA, root finding, optimization

Motivating example: Maximum Likelihood

$X_1, \dots, X_n \sim$ IID $N(\mu, 1)$ observations

$$\mathcal{L}(\mu) = \prod_1^n f(x_i; \mu)$$

$$= \prod_1^n \frac{1}{\sqrt{2\pi}} e^{-(x_i-\mu)^2/2}$$

$$= \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2}\sum (x_i-\mu)^2}$$

$$\ell(\mu) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\sum_1^n (x_i-\mu)^2$$

The MLE of μ, $\hat{\mu}$, satisfies $\underbrace{\ell'(\mu) = 0}_{\text{root finding on } \ell'}$, maximization on ℓ.

<u>In this case</u> $\hat{\mu}$ can be found analytically:

$$\ell'(\mu) = -\sum (x_i - \mu)$$

$$\Rightarrow \ell'(\hat{\mu}) = 0 \quad \Rightarrow \quad \hat{\mu} = \frac{1}{n}\sum_1^n x_i$$

<u>In general</u>, ℓ, ℓ', etc cannot be evaluated analytically, particularly when multiple parameters.

Root finding i.e. solving $f(x) = 0$

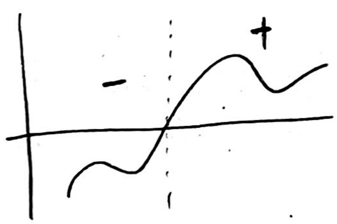
in one-dimension

Two basic ideas:

- search using values of f
- use derivative info (or approximate derivative info)

Bisection

Idea: Look for sign changes of f (only $\mathrm{sgn}(f(x))$ is used)



- Start with interval $[a_0, b_0]$ which contains a sign change.
- Set $x_0 = \dfrac{a_0 + b_0}{2}$
- If $f(a_0) f(x_0) < 0$ set $a_1 = a_0$, $b_1 = x_0$

  If $f(x_0) f(b_0) < 0$ set $a_1 = x_0$, $b_1 = b_0$

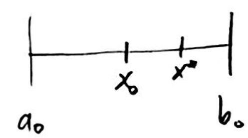
- Repeat to compute $[a_n, b_n]$
- Take $x_n = \dfrac{a_n + b_n}{2}$ as estimate for root of $f(x)$.

Rate of Convergence

- If $[a_0, b_0]$ contains a sign change, then

if x^* is true root, $x_0 = \frac{a_0+b_0}{2}$ satisfies

$$|x_0 - x^*| \le \frac{b_0 - a_0}{2}$$



Subsequently $\underbrace{|x_n - x^*|}_{\epsilon_n} \le \frac{b_0 - a_0}{2^{n+1}}$.

linear convergence

$$\frac{\epsilon_{n+1}}{\epsilon_n} = \frac{b_0-a_0}{2^{n+2}} \frac{2^{n+1}}{b_0-a_0} = \frac{1}{2}$$

If we require $|x_n - x^*| \le \epsilon$, then we need

$$\frac{b_0-a_0}{2^{n+1}} \le \epsilon \quad \text{(solve for } n\text{)}$$

$$\Rightarrow \log_2 \frac{b_0 - a_0}{\epsilon} \le n+1$$

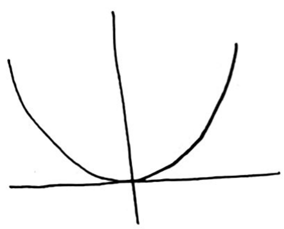
$$\Rightarrow n \ge \log_2 \frac{b_0 - a_0}{\epsilon} - 1$$

I.e. $\epsilon \approx 10^{-3} \Rightarrow n \approx 10$ if $b_0 - a_0 \approx 1$.

Main Failure Modes of Bisection

There is no failure mode! So long as $[a_0, b_0]$ contains a sign change. Bullet-proof

Note Bisection is not applicable to $f(x) = x^2$, for example.



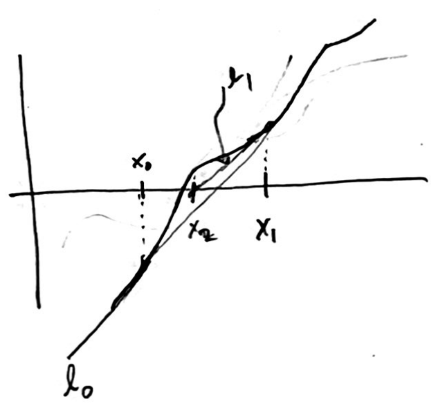
Root, but no sign change.

Newton's Method

Bisection only used function value information, how do we incorporate derivative information?

Idea Approximate f as a linear function (value + slope) and find root of the approximation – use this as estimate of true root, repeat.

Graphically:



What is the algorithm? Given initial guess x_0, evaluate $f(x_0)$ and $f'(x_0)$. Form approximation near x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = l_0(x)$$

At the root of l_0, call it x_1, $l_0(x_1) = 0$.

$$\Rightarrow \quad f(x_0) + f'(x_0)(x_1 - x_0) = 0$$

$$\Rightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

and repeat to obtain:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \Big\} \quad \underline{\text{Newton's Method}}$$

<u>Error Analysis</u>

<u>Assuming</u> Newton's Method converges, (why wouldn't it converge?)

how fast does it converge?

<u>Taylor's Thm</u> says that (if f, f', f'' exist and are continuous near x_0)

$$\underset{\substack{\uparrow \\ \underline{\text{equals}}}}{f(x) =} f(x_0) + f'(x_0)(x - x_0) + \frac{f''(\xi)(x - x_0)^2}{2} \quad \text{for some } \xi \in [x_0, x]$$

Evaluating this expression at x^*, the true root, gives:

$$(*) \qquad f(x^*) = 0 = f(x_0) + f'(x_0)(x^* - x_0) + \frac{f''(\xi)}{2!}(x^* - x_0)^2$$

And Newton's Method says that

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_1)}$$

We are interested in the absolute error $|x^* - x_1|$

$$(**) \qquad \Rightarrow \quad x^* - x_1 = x^* - x_0 + \frac{f(x_0)}{f'(x_0)}$$

From Taylor, $(*)$ $\quad x^* - x_0 = -\dfrac{f(x_0)}{f'(x_0)} - \dfrac{f''(\xi)}{2f'(x_0)}(x^* - x_0)^2$

Inserting into $(**)$ we have

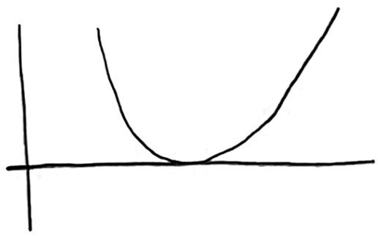
$$x^* - x_1 = -\frac{f''(\xi)\,(x^* - x_0)^2}{2\,f'(x_0)}$$

$$\Rightarrow |x^* - x_1| = \varepsilon_1 = \frac{1}{2}\left|\frac{f''(\xi)}{f'(x_0)}\right| |x^* - x_0|^2 = \frac{1}{2}\overbrace{\left|\frac{f''(\xi)}{f'(x_0)}\right|}^{\text{Basically requires this to be bounded.}} \varepsilon_0^2$$

So $\quad \dfrac{\varepsilon_{n+1}}{\varepsilon_n^2} = \dfrac{1}{2}\left|\dfrac{f''(\xi)}{f'(x_0)}\right|$ $\qquad$ <u>Quadratic Convergence</u>

Very fast, very rare - hence why Newton's Method is so powerful.

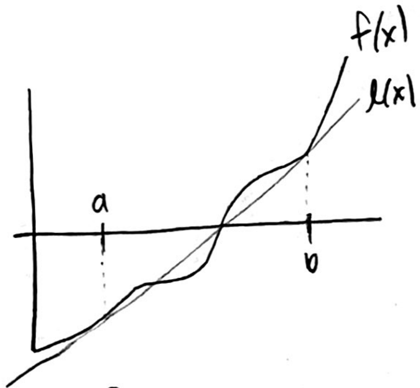
Failure Modes of Newtons Method



① Repeated roots

Newton's Method will "sort of" converge, but not quadratically

Example



② Oscillating behavior

Happens rarely, but happens because x_0 is not close enough to root for Newton to converge.

Quick programming demo

Quasi-Newton Methods

Newton : $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$

→ Replace with an approximation, e.g.

$$\left. f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \right\} \Rightarrow \text{Secant Method}$$

Multivariate Newton's Method

Newton's Method can be generalized to systems of non-linear equations:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned} \quad \Longleftrightarrow \quad \underbrace{\vec{f}(\vec{x}) = \vec{0}}_{\text{Vector notation}}.$$

An analogous Taylor approximation can be made:

$$(***) \qquad \vec{f}(\vec{x}) \approx \vec{f}(\vec{x}_0) + \bar{J}(\vec{x}_0)(\vec{x} - \vec{x}_0)$$

where J is the Jacobian matrix:

$$J_{ij}(\vec{x}_0) = \frac{\partial f_i}{\partial x_j}(\vec{x}_0)$$

Assuming $J^{-1}(\vec{x}_0)$ exists, the multivariate Newton's method is given by: (by evaluating $(***)$ at true root $\vec{x}^*$)

$$\vec{x}_{n+1} = \vec{x}_n - \underbrace{J^{-1}(\vec{x}_0)}_{\substack{n\times n \\ \text{matrix}}} \underbrace{\vec{f}(\vec{x}_0)}_{\substack{n\times 1 \\ \text{vector.}}}$$

Convergence can also be shown to be quadratic:

$$\| \vec{x}_{n+1} - \vec{x}^* \| \approx C \| \vec{x}_n - \vec{x}^* \|^2.$$

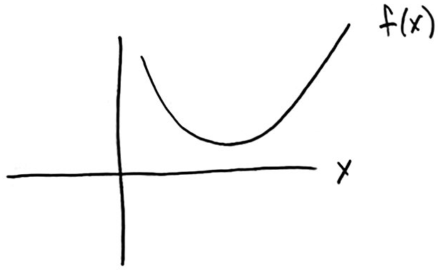
Optimization

There are two basic ways to classify optimization problems:

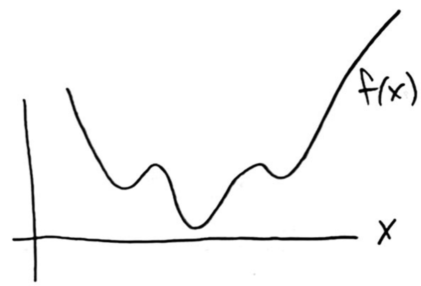
convex vs. non-convex

unconstrained vs. constrained

In one-dimension:



convex



non-convex

EASY

$f'(x)=0$ has one unique solution

HARD hard to design algorithms that do not get "stuck" in local minima

Eg. $f'(x)=0$ has many solutions

Unconstrained

$$\max_{x \in \mathbb{R}} f(x)$$

Constrained

$$\max_{x} f(x)$$

subject to $g(x) = 0$. (or $\leq$)

Example

$$\max_{x} e^x$$

such that $|x-5| \leq 0$

We will only talk about methods for convex unconstrained problems. (See dedicated optimization courses at Courant for more.)

Comparison with Root finding

$\max_{\vec{x}} f(\vec{x})$ ← notice f is scalar-valued

At the maximum, $\nabla f(\vec{x}^*) = \vec{0}$ — so we must find roots of $\vec{\nabla f} = \vec{0}$.

Simplest method: Gradient Descent (Ascent)

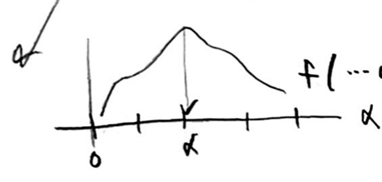
∇f points in the direction of steepest change of the function f — starting at $\vec{x}_0$, move in this direction to obtain next estimate of maximum:

$$\vec{x}_1 = \vec{x}_0 + \alpha \nabla f(\vec{x}_0)$$

↖ choose α so that $f(\vec{x}_1) \geq f(\vec{x}_0)$

Most elementary method: line search

choose α to approximately maximize $\underbrace{f(\vec{x}_0 + \alpha \nabla f(\vec{x}_0))}_{\text{function of one variable}}$



For convex problems, and proper choice of α, this method is linearly convergent (like bisection).

Values of ∇f are used to solve $\nabla f = \vec{0}$, but not derivatives of ∇f. (Foreshadow to stochastic gradient descent)

A Newton Method

Newton's method generalizes straightforwardly to optimization, except we need 2nd partial derivatives of f.

Recall, we are trying to find a zero of ∇f, so approx by a Taylor series:

$$\nabla f(\vec{x}) \approx \nabla f(\vec{x}_0) + H(\vec{x}_0)(\vec{x} - \vec{x}_0)$$

where $\nabla f = \begin{pmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_n \end{pmatrix}$ and $\underbrace{H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}}$

this is known as the Hessian matrix : symmetric

↓

what else?

$\Rightarrow$ the Newton iteration is then

$$\vec{x}_{n+1} = x_n - H^{-1}(\vec{x}_n)\,\nabla f(\vec{x}_n)$$

] computational complexity?

Often H cannot be computed, (for whatever reason), or it is too expensive to compute. Approximating it by $\tilde{H}$ gives us a "Quasi-Newton method".

→ Mention step-size control.

→ Mention General form of optimization algorithms : $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$