Finding gradients by hand

Friday, April 9, 2021 6:37 AM

Overview: how to find gradients?

1. Don't find them.... use derivative-free methods "DFO"
   =Deriv.Free Optim.

These are usually better than finite-diff. methods,
at least if function evaluations are expensive.

Work fine in 1D, OK in small dim., horrible in high dimensions

Usually reserved for last-resort, gradient doesn't exist or hard to obtain

Algos

- old-school stuff like Nelder-Mead simplex-reflection ≠ simplex (LP) method
  slow, no guarantees
- coordinate descent, pattern search
  also slow, not state-of-the-art

} Matlab/scipy has these. Usually not competitive

- implicit filtering
  like finite-diff. but w/o a tiny stepsize h for gradient

- model-based
  - Polynomial models (linear or quadratic)
    w/ trust-region
    State-of-the-art. Scales to ≈1000 dimensions
    See Nocedal + Wright or
    "Intro to DFO" (SIAM '09) Conn, Scheinberg, Vicente
  - Gaussian Process model
    w/ Acquisition Function "Bayesian Optimization"
    trades off exploration / exploitation
    Handles discrete variables and noisy functions
    (so used often for hyperparameter tuning)
    BoTorch software
    poor above ≈20 dim

- others

- hybrid model/heuristic: CMA-ES
- evolutionary search, "zeroth order" methods, Spall's SPSA, Becker's SSD
- heuristics: particle-swarm, etc

Often applied to nonconvex problems so bad stationary pts an issue

② Finite difference computation to approximate gradient

Usually not the best solution (i.e. poor "long-term" solution) but not to be overlooked, especially for prototyping & back-of-envelope simulations.

Better (i.e. cheaper) in small dimensions

Fundamentally low accuracy

Forward diff $f: \mathbb{R}^n \to \mathbb{R}$ so $\nabla f(x) \in \mathbb{R}^n$

canonical unit basis vector

$$\forall i \in [n], \quad (\nabla f(x))_i \simeq \frac{f(x+he_i)-f(x)}{h}$$

so n+1 function evaluations

$h \approx 10^{-8}$ rule-of-thumb: too large and Taylor Series is inaccurate
too small and roundoff makes it less accurate

Centered diff is better (keeps about 2/3 of floating pt. digits)
but needs 2n function evals

$$\simeq \frac{f(x+he_i)-f(x-he_i)}{2h}$$

Neither works if f is noisy

③ Compute gradient by hand (or via Mathematica, etc.)

- Product Rule $\quad (fg)' = f'g + f \cdot g'$
- Leibniz Rule: functions defined by integrals
- Functions defined as min/max

see later set of notes

Chain Rules

simple 1D $\quad \frac{d}{dx}\, g \circ f\,(x) = g'(f(x)) \cdot f'(x)$

multi-variate
$$\underbrace{\mathbb{R}^n \xrightarrow{f} \mathbb{R}^m \xrightarrow{g} \mathbb{R}^p}_{h = g \circ f}$$

Def Jacobian $\quad (Jf)(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x) \quad$ i.e. if $f: \mathbb{R}^n \to \mathbb{R}^1$ then $(Jf)(x) = \nabla f(x)^T$

then

$$(Jh)(x) = (Jg)(f(x)) \cdot (Jf)(x)$$

matrix multiplication

special cases: $p=1$ so $Jh = \nabla h^T$

then $\quad \nabla h(x) = (Jf)(x)^T \cdot \nabla g(f(x))$

and if $f(x) := Ax - b$ this simplifies to

$$\nabla h(x) = A^T \cdot \nabla g(Ax - b)$$

total derivative version

let $z = f(x,y)$ and $x = g(t)$, $y = h(t)$

then $\quad \frac{dz}{dt} = \frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt}$

i.e., implicit differentiation

under the conditions of implicit fcn thm

$F(x,y) = 0$ implicitly defines a function $y = f(x)$ (so $F(x, f(x)) = 0$)

since $F(x,y) = 0$

$\Rightarrow \frac{dF}{dx}(x,y) = 0$, so chain rule:

$$0 = \frac{dF}{dx} = \underbrace{\frac{\partial F}{\partial x}}_{F_x} \cdot \overset{=1}{\cancel{\frac{dx}{dx}}} + \underbrace{\frac{\partial F}{\partial y}}_{F_y} \cdot \frac{dy}{dx}$$

i.e. $\quad \frac{dy}{dx} = \frac{-F_x}{F_y}$

Examples

Scalar cases usually straightforward
vector/matrices more work.
See Boyd + Vandenberghe examples such as $\nabla \log\det(X)$
or Peterson + Pederson's "Matrix Cookbook"

or use Kronecker product tricks

$(B^T \otimes A)\,\text{vec}(X) = \text{vec}(A \cdot X \cdot B)$

ex: $f(U) = \frac{1}{2}\| U \cdot V^T - B \|_F^2$, $U \in \mathbb{R}^{m \times r}$ is a matrix

idea:
expand

$\langle A, B \rangle = tr(A^T B)$

$$f(U+\Delta) = f(U) + \langle \underbrace{\nabla f(U)}_{\text{find this term}}, \Delta \rangle + O(\|\Delta\|^2)$$

$$\begin{aligned} f(U+\Delta) &= \tfrac{1}{2}\| (U+\Delta)V^T - B \|_F^2 \\ &= \tfrac{1}{2}\| (UV^T - B) + \Delta \cdot V^T \|_F^2 \\ &= \underbrace{\tfrac{1}{2}\| UV^T - B \|_F^2}_{f(U)\ \checkmark} + \langle UV^T - B, \Delta \cdot V^T \rangle + \underbrace{\tfrac{1}{2}\| \Delta V^T \|_F^2}_{O(\|\Delta\|^2)\ \checkmark} \end{aligned}$$

$$\begin{aligned} &= tr\left((UV^T - B)^T \cdot \Delta \cdot V^T \right) \\ &= tr\left(V^T (UV^T - B)^T \cdot \Delta \right) \qquad \text{via cyclic property} \\ &= tr\left(((UV^T - B) \cdot V)^T \cdot \Delta \right) \\ &= \langle \underbrace{(UV^T - B) \cdot V}_{\text{so this is the gradient}}, \Delta \rangle \end{aligned}$$

i.e. $\nabla_U \frac{1}{2}\| UV^T - B \|_F^2 = (UV^T - B) \cdot V$

as you'd expect via chain rule, only being careful since matrix multiplication isn't commutative.