Section 2: Optimization

Ruofan Ma

Gov2018 2024 Spring

February 7, 2024

Motivation

- Why is optimization relevant?
  - Regression as minimizing the loss function (OLS):

$$\hat{\beta}_{\mathrm{OLS}} = \arg\min_{\beta} \left(\frac{1}{2} \sum_{n=1}^{N} \left(y_n - \beta^\top \mathbf{x}_n \right)^2 \right)$$

  - Similarly, MLE (Olivella, Pratt, and Imai, 2022):

$$\begin{aligned} & P(\mathbf{Y}, \mathbf{L} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{B}, \mathbf{X}) \\ & \propto \prod_{m=1}^{M} \left[\frac{\Gamma(M\eta)}{\Gamma(M\eta + U_{m\cdot})} \prod_{n=1}^{M} \frac{\Gamma(\eta + U_{mn})}{\Gamma(\eta)} \right] \\ & \times P(\mathbf{s}_1) \prod_{t=2}^{T} \prod_{m=1}^{M} \prod_{it \in V_t} \left[\frac{\Gamma(\alpha_{it\cdot m})}{\Gamma(\alpha_{it\cdot m} + 2N_t)} \prod_{k=1}^{K} \frac{\Gamma(\alpha_{itmk} + C_{itk})}{\Gamma(\alpha_{itmk})} \right]^{I(S_t = m)} \\ & \times \prod_{t=1}^{T} \prod_{p,q \in V_t} \prod_{g,h=1}^{K} \left(\theta_{pqtgh}^{y_{pqt}} (1 - \theta_{pqtgh})^{1 - y_{pqt}} \right)^{z_{p \to q,t,g} \times w_{q \leftarrow p,t,h}} \end{aligned}$$

Figure: A likelihood function that you might end up working with

Roadmap for today

- Gradient Descent: Generalization of univariate differentiation
- Newton-Raphson: Root-searching Algorithm
- Difference between GD and NR

- Some coding exercise (`Rmd` file available on course website)

Gradient Descent: Basic Setup

- Geometric intuition behind GD: For one-dimensional x, the line that passes through a given point $(x^t, f(x^t))$ with a given slope $f'(x^t)$ can be written as

$$y - f\left(x^t\right) = f'\left(x^t\right)\left(x - x^t\right)$$

- Generalizing this to the multi-dimensional case, we have:

$$y - f\left(\mathbf{x}^t\right) = \left(\nabla f\left(\mathbf{x}^t\right)\right)^\top\left(\mathbf{x} - \mathbf{x}^t\right) \equiv \left\langle \mathbf{x} - \mathbf{x}^t, \nabla f\left(\mathbf{x}^t\right)\right\rangle$$

which can also be written as $y = f(\mathbf{x}^t) + (\nabla f(\mathbf{x}^t))^\top (\mathbf{x} - \mathbf{x}^t)$

Gradient Descent: Formalization

- Gradient descent/ascent (Cauchy, 1947) for **mode finding**:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} \pm \eta \nabla f\left(\mathbf{x}^{(t)}\right)$$

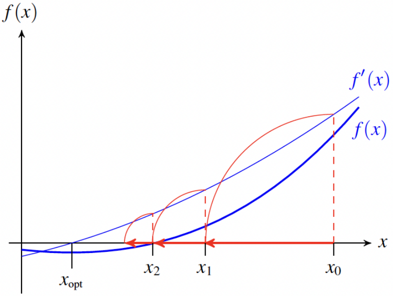
  , where η is called the "learning rate". A key challenge is to choose η properly, and perhaps adaptively. Selecting too small of an η can make the algorithm too slow, and too large the algorithm can be too unstable!
- $+/-$: Recall that the gradient vector points in the direction of steepest **ascent** in $f(\mathbf{x})$. So when doing gradient ascent, we take the plus sign to indicate a step **towards** the next larger value; and conversely, we take the minus sign when doing gradient descent.

Gradient Descent: An One-Dimensional Example

- Algorithm: Gradient Descent (searches for a minimum of $f(\cdot)$)
  1. Start with some point $x \in \mathbb{R}$ and fix a precision $\varepsilon > 0$
  2. Repeat for $n = 1, 2, \cdots$:

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

  3. Terminate when $|f'(x_n)| < \varepsilon$



Netwon-Raphson: Basic Setup

- Note that finding a mode of $f(x)$ is (almost) equivalent to finding a root of $f'(x)$. So alternative to GD, we can also try to find the solution of $0 = g(x) \triangleq f'(x)$, which, in 1D, gives:

$$x^{(t+1)} = x^{(t)} - \frac{g\left(x^{(t)}\right)}{g'\left(x^{(t)}\right)} \equiv x^{(t)} - \frac{f'\left(x^{(t)}\right)}{f''\left(x^{(t)}\right)}$$

- Multi-dimensional:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{\mathbf{g}\left(\mathbf{x}^{(t)}\right)}{J\left(\mathbf{x}^{(t)}\right)} \equiv \mathbf{x}^{(t)} - \frac{\nabla f\left(\mathbf{x}^{(t)}\right)}{\nabla^2 f\left(\mathbf{x}^{(t)}\right)}$$

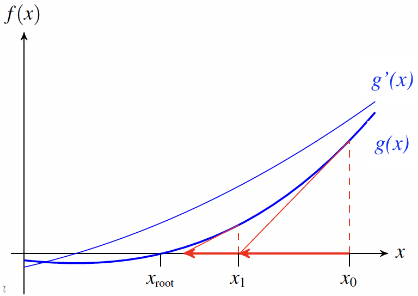
where $J(\mathbf{x})$ is the Jacobian matrix (Hessian of function $f(\mathbf{x})$)

Netwon-Raphson: An One-Dimensional Example

- Algorithm: NR method (root searching)
  1. Start with some point $x \in \mathbb{R}$ and fix a precision $\varepsilon > 0$
  2. Repeat for $n = 1, 2, \cdots$

$$x_{n+1} := x_n - f'(x_n) / f''(x_n)$$

  3. Terminate when $|f'(x_n)| < \varepsilon$



GD vs. NR: Which one to use? (1)

- Consider, for example, the case of OLS, whose target function (the function to be minimized) can be written as:

$$g(\boldsymbol{\beta}) = \frac{1}{2}\|\underbrace{X}_{n\times p}\underbrace{\boldsymbol{\beta}}_{p\times 1} - \underbrace{\mathbf{y}}_{n\times 1}\|_2^2$$

  - Gradient: $\nabla g(\boldsymbol{\beta}) = X^\top(X\boldsymbol{\beta} - \mathbf{y})$; Hessian $\nabla^2 g(\boldsymbol{\beta}) = X^\top X$
  - GD Update: $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \eta X^\top \left(X\beta^{(t)} - \mathbf{y}\right)$
  - Newton-Raphson:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \left(X^\top X\right)^{-1} X^\top \left(X\boldsymbol{\beta}^{(t)} - \mathbf{y}\right) = \left(X^\top X\right)^{-1} X^\top \mathbf{y}$$

  - In other words, NR converges in one step to the OLS solution.

GD vs. NR: Which one to use? (2)

- Consider now, the case of logistic regression:

$$y_i \overset{\text{iid}}{\sim} \text{Bern}(\theta_i)\,; \theta_i = \frac{\exp\left(\mathbf{x}_i^\top \boldsymbol{\beta}\right)}{1+\exp\left(\mathbf{x}_i^\top \boldsymbol{\beta}\right)}, \mathbf{x}_i = \left(x_{i1}, \ldots, x_{ip}\right)^\top$$

- Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^\top, X_{n\times p} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$, and $\mathbf{y} = (y_1, \ldots, y_n)^\top$
- Log-likelihood: $\ell(\boldsymbol{\beta}) = \sum_{i=1}^n \left[y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \log\left(1+\exp\left(\mathbf{x}_i^\top \boldsymbol{\beta}\right)\right)\right]$
- Gradient:

$$\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^n \left(y_i \mathbf{x}_i - \frac{\exp\left(\mathbf{x}_i^\top \boldsymbol{\beta}\right)}{1+\exp\left(\mathbf{x}_i^\top \boldsymbol{\beta}\right)} \mathbf{x}_i\right)$$

$$= \sum_{i=1}^n (y_i - \theta_i)\,\mathbf{x}_i = X^\top(\mathbf{y} - \boldsymbol{\theta})$$

GD vs. NR: Which one to use? (3)

- Hessian matrix:

$$H(\boldsymbol{\beta}) = -X^\top \left(\nabla_{\boldsymbol{\beta}} \boldsymbol{\theta}\right) = \sum_{i=1}^{n} \mathbf{x}_i \theta_i \left(1 - \theta_i\right) \mathbf{x}_i^\top \equiv X^\top D_w X$$

- GD moves: $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \epsilon X^\top \left(\mathbf{y} - \boldsymbol{\theta}^{(t)}\right)$
- NR moves: $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \left(X^\top D_w X\right)^{-1} X^\top \left(\mathbf{y} - \boldsymbol{\theta}^{(t)}\right)$
  - which can be understood as iterative weighted least squares.
- High-level takeaway: Compared to GD, NR converges very fast when it works, but may be unstable and the involved computation is heavy.