EC708 Discussion 8
Numerical Optimization

Yan Liu[1]

Department of Economics
Boston University

March 18, 2022

[1]Parts of the materials are from Train (2009), Kochenderfer and Wheeler (2019), and teaching slides of Shuowen Chen, Jean-Jacques Forneron, and Ryan Tibshirani.

Outline

Optimization Problem

MLE

Consider maximizing the log-likelihood function

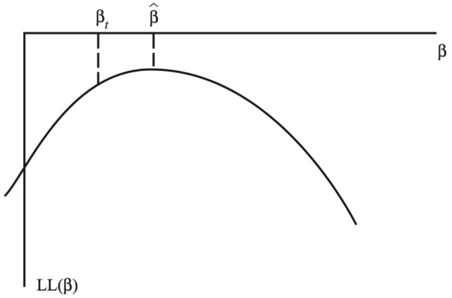
$$\bar{\ell}_T(\theta) = \sum_{t=1}^{T} \ell_t(\theta),$$

where $\theta \in \Theta \subset \mathbb{R}^K$.

- **Goal:** find $\hat{\theta}_T = \arg\max_{\theta \in \Theta} \bar{\ell}_T(\theta)$.
- To utilize minimization packages in practice, we usually work with $-\bar{\ell}_T(\theta)$.

Optimization Problem

Graphical Illustration of MLE



Finding $\hat{\theta}_T$ is a hill-climbing process:

1. Specify a starting value θ_0.
2. Each step k moves to a new value θ_{k+1} at which $\bar{\ell}_T(\theta)$ is higher than at the current value θ_k.
3. Keep climbing until no further increase can be found.

Optimization Problem

Gradient and Hessian

Gradient in step k:

$$g_k = \left(\frac{\partial \bar{\ell}_T(\theta)}{\partial \theta} \right)_{\theta_k}.$$

Hessian in step k:

$$H_k = \left(\frac{\partial g_k}{\partial \theta'} \right)_{\theta_k} = \left(\frac{\partial^2 \bar{\ell}_T(\theta)}{\partial \theta \partial \theta'} \right)_{\theta_k}.$$

The gradient tells use in what direction to climb, and the Hessian can help us to know how far to climb.

Optimization Problem

Numerical Evaluation of Derivatives

Recall the definition of first-order partial derivative of function f at θ:

$$\frac{\partial f(\theta)}{\partial \theta_j} = \lim_{h \to 0} \frac{f(\theta_1, \ldots, \theta_j + h, \ldots, \theta_K) - f(\theta_1, \ldots, \theta_j, \ldots, \theta_K)}{h}$$

for $j = 1, \ldots, K$. Numerically, we can approximate it by calculating

$$f_j(\theta) = \frac{f(\theta + he_j) - f(\theta)}{h},$$

where $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the unit vector with 1 in position j and h is the step size. In practice, a more accurate approximation is

$$f_j(\theta) = \frac{f(\theta + he_j) - f(\theta - he_j)}{2h}.$$

Outline

Newton-Raphson Method

Algorithm

Take a second-order Taylor's approximation of $\bar{\ell}_T(\theta_{k+1})$ around $\bar{\ell}_T(\theta_k)$:

$$\bar{\ell}_T(\theta_{k+1}) = \bar{\ell}_T(\theta_k) + (\theta_{k+1} - \theta_k)' g_k + \frac{1}{2}(\theta_{k+1} - \theta_k)' H_k (\theta_{k+1} - \theta_k).$$

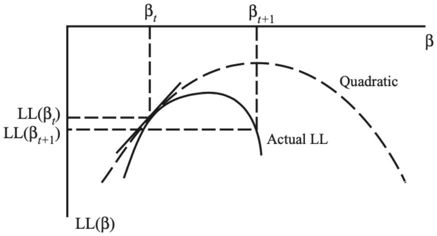
Find θ_{k+1} that maximizes this approximation:

$$g_k + H_k(\theta_{k+1} - \theta_k) = 0 \;\Rightarrow\; \theta_{k+1} = \theta_k - \underbrace{H_k^{-1}}_{\text{step size}} \cdot \underbrace{g_k}_{\text{direction}} .$$

- Iterate until convergence, which can be defined in many ways:
  - $\bar{\ell}_T(\theta_{k+1})$ close to $\bar{\ell}_T(\theta_k)$
  - θ_{k+1} close to θ_k
  - g_{k+1} close to g_k
- If $\bar{\ell}_T(\theta)$ were exactly quadratic, then Newton-Raphson reaches the maximum in one step from any starting value.

Newton-Raphson Method

Step Size

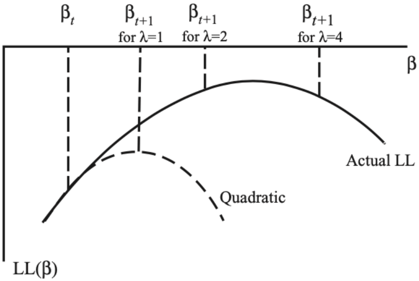


- It is possible for Newton-Raphson to step past the maximum and move to a lower $\bar{\ell}_T(\theta)$.
- To ensure each step provides an increase in $\bar{\ell}_T(\theta)$, we introduce a scalar step size λ_k:

$$\theta_{k+1} = \theta_k - \lambda_k H_k^{-1} g_k.$$

Newton-Raphson Method

Determining the Step Size: Backtracking Line Search



Perform step-size adjustment in each step. Start with $\lambda_k = 1$.

- If $\bar{\ell}_T(\theta_{k+1}) < \bar{\ell}_T(\theta_k)$, continue halving λ_k until $\bar{\ell}_T(\theta_{k+1}) > \bar{\ell}_T(\theta_k)$.
  - A tiny λ_k is a signal that a different iteration procedure is needed.
- If $\bar{\ell}_T(\theta_{k+1}) > \bar{\ell}_T(\theta_k)$, continue doubling λ_k as long as doing so further raises $\bar{\ell}_T(\theta_{k+1})$.
  - Raising λ_k reduces # of iterations needed to reach the maximum.

Newton-Raphson Method

Drawbacks

- Calculations of the Hessian is usually computation-intensive.
  - $\frac{K(K+1)}{2}$ functions to evaluate in each step
  - Numerically calculated Hessian might be ill-behaved (singular).
- Does not guarantee an increase in each step if the log-likelihood function is not globally concave.
  - Hessian may not be negative definite.
  - Remedy: regularization. Instead of using H_k^{-1} directly, use

$$(H_k + \mu_k I_K)^{-1}$$

where $\mu_k < 0$ guarantees negative definiteness.

Gauss-Newton Method

Consider the following general nonlinear model:

$$y_t = f(x_t;\theta) + u_t, \quad t = 1, \ldots, T$$

- $u_t \sim$ i.i.d. $N(0, \sigma^2)$. Assume σ^2 is known here.
- x_t: $M \times 1$ exogenous regressors.
- $f(\cdot)$: some function satisfying some regularity conditions.

Gauss-Newton Method

Due to the normality assumption, we have the log-likelihood function

$$\bar{\ell}_T(\theta) = -\frac{T}{2}\ln 2\pi - \frac{T}{2}\ln\sigma^2 - \underbrace{\sum_{t=1}^{T}(y_t - f(x_t;\theta))^2}_{\equiv S(\theta)}.$$

It suffices to work with $S(\theta)$. Its first and second derivatives w.r.t. θ are

$$g(\theta) = 2\sum_{t=1}^{T}\frac{\partial u_t}{\partial\theta}u_t, \quad H(\theta) = 2\sum_{t=1}^{T}\left[\frac{\partial u_t}{\partial\theta}\frac{\partial u_t}{\partial\theta'} + \frac{\partial^2 u_t}{\partial\theta\partial\theta'}u_t\right].$$

In $H(\theta)$, the blue term is usually small relative to the red term, so we neglect it and use the following:

$$\theta_{k+1} = \theta_k - \left[\sum_{t=1}^{T}\frac{\partial u_t}{\partial\theta}\frac{\partial u_t}{\partial\theta'}\right]^{-1}\Bigg|_{\theta=\theta_k}\sum_{t=1}^{T}\frac{\partial u_t}{\partial\theta}u_t\Bigg|_{\theta=\theta_k}.$$

Gauss-Newton Method

Remarks:

- This method doesn't compute second-order derivatives.
- Has an OLS interpretation. Let $z_t = -\partial u_t / \partial \theta$, then

$$\theta_{k+1} = \theta_k + \left(\sum_{t=1}^{T} z_t z_t' \right)^{-1} \Bigg|_{\theta=\theta_k} \sum_{t=1}^{T} z_t u_t \Bigg|_{\theta=\theta_k}.$$

- Similar regularization can be incorporated as in Newton-Raphson: Marquart quadratic hill climbing

$$\theta_{k+1} = \theta_k + \left(\sum_{t=1}^{T} z_t z_t' + \mu I_K \right)^{-1} \Bigg|_{\theta=\theta_k} \sum_{t=1}^{T} z_t u_t \Bigg|_{\theta=\theta_k}.$$

Outline

Quasi-Newton Methods

- Both main steps in Newton-Raphson method could be expensive:
  - Compute Hessian H_k
  - Solve the system $H_k\Delta = -g_k$
- Quasi-Newton methods repeat updates of the form

$$\theta_{k+1} = \theta_k - \lambda_k {G_k}^{-1} g_k$$

for some approximation G_k of H_k. We want G_k to be easy to compute and linear system $G_k\Delta = -g_k$ to be easy to solve.

Quasi-Newton Methods

Berndt-Hall-Hall-Hausman (BHHH)

Berndt et al. (1974) utilize the fact that the objective function is the sum of log likelihoods and propose to use scores to approximate Hessian.

- Score of observation t:

$$s_t(\theta_k) = \frac{\partial \ell_t(\theta)}{\partial \theta}\Big|_{\theta=\theta_k}.$$

- Gradient is the sum of scores:

$$g_k = \sum_{t=1}^{T} s_t(\theta_k).$$

Quasi-Newton Methods

Berndt-Hall-Hall-Hausman (BHHH)

- Outer product of observation t's score is the $K \times K$ matrix

$$s_t(\theta_k)s_t(\theta_k)' = \begin{pmatrix} s_t^1 s_t^1 & s_t^1 s_t^2 & \cdots & s_t^1 s_t^K \\ s_t^2 s_t^1 & s_t^2 s_t^2 & \cdots & s_t^2 s_t^K \\ \vdots & \vdots & & \vdots \\ s_t^K s_t^1 & s_t^K s_t^2 & \cdots & s_t^K s_t^K \end{pmatrix},$$

  where s_t^j is the j-th element of $s_t(\theta_k)$.

- Sum of outer product of scores:

$$G_k = \sum_{t=1}^{T} s_t(\theta_k)s_t(\theta_k)'.$$

Berndt-Hall-Hall-Hausman (BHHH) update uses G_k in place of $-H_k$:

$$\theta_{k+1} = \theta_k + \lambda_k G_k^{-1} g_k.$$

Quasi-Newton Methods

Berndt-Hall-Hall-Hausman (BHHH)

Why does BHHH work?

- At maximum, G_k is the sample variance of scores and thus provides a measure of the log-likelihood functions' curvature, similar to H_k.
- These ideas are formalized in the information matrix equality.
- G_k is far faster to calculate than H_k and necessarily positive definite.

Drawbacks: BHHH can give small steps when far from the maximum because G_k is not a good approximation to $-H_k$.

Quasi-Newton Methods

BFGS and DFP

- BHHH uses only information at θ_k to determine each step.
- As G_k already contains information about the Hessian, BFGS and DFP use suitable matrix update to form G_{k+1}.
- General procedure: In each iteration k,
  1. Compute Quasi-Newton direction $\Delta_k = -G_k^{-1} g_k$
  2. Determine stepsize λ_k (by backtracking line search)
  3. Update $\theta_{k+1} = \theta_k + \lambda_k \Delta_k$
  4. Compute G_{k+1} from G_k

Quasi-Newton Methods

BFGS and DFP

Reasonable requirement for G_{k+1} (motivated by secant method):

$$g_{k+1} = g_k + G_{k+1}\Delta_k.$$

In addition, we want:

- G_{k+1} to be symmetric
- G_{k+1} to preserve positive definiteness (BFGS and DFP deal with convex optimization)

Quasi-Newton Methods

Broyden-Fletcher-Goldfarb-Shanno (BFGS) update:

BFGS uses a rank-two update of the form:

$$G_{k+1} = G_k + auu' + bvv'.$$

Let $\gamma_k = g_{k+1} - g_k$. The secant equation yields

$$\gamma_k - G_k \Delta_k = (au'\Delta_k)u + (bv'\Delta_k)v.$$

Putting $u = \gamma_k$, $v = G_k \Delta_k$, and solving for a, b, we get

$$G_{k+1} = G_k - \frac{G_k \Delta_k \Delta_k' G_k}{\Delta_k' G_k \Delta_k} + \frac{\gamma_k \gamma_k'}{\gamma_k' \Delta_k}.$$

- BFGS update is quite cheap: $O(K^2)$ operations
- BFGS is the algorithm behind Matlab's fminunc.

Quasi-Newton Methods

Davidon-Fletcher-Powell (DFP)

Davidon-Fletcher-Powell (DFP) update:

DFP pursues the same idea to update G_{k+1}^{-1}:

$$G_{k+1}^{-1} = G_k^{-1} + auu' + bvv'.$$

The secant equation yields

$$\Delta_k - G_k^{-1}\gamma_k = (au'\gamma_k)u + (bv'\gamma_k)v.$$

Putting $u = \Delta_k$, $v = G_k^{-1}\gamma_k$, and solving for a, b, we get

$$G_{k+1}^{-1} = G_k^{-1} - \frac{G_k^{-1}\gamma_k\gamma_k' G_k^{-1}}{\gamma_k' G_k^{-1}\gamma_k} + \frac{\Delta_k\Delta_k'}{\Delta_k'\gamma_k}.$$

- The role of γ_k and Δ_k is swapped.
- DFP is not as popular as BFGS. There is some evidence that BFGS is more efficient than DFP.

Outline

Steepest Ascent

The greatest possible increase in $\bar{\ell}_T(\theta)$ for the (small enough) distance between θ_k and θ_{k+1} is provided by

$$\theta_{k+1} = \theta_k + \lambda_k g_k.$$

Motivated by the Lagrangian:

$$L = \underbrace{\bar{\ell}_T(\theta_k) + (\theta_{k+1} - \theta_k) g_k}_{\text{1st-order Taylor expansion of } \bar{\ell}_T(\theta_{k+1})} - \frac{1}{2\lambda_k} \underbrace{[(\theta_{k+1} - \theta_k)'(\theta_{k+1} - \theta_k) - d]}_{\text{distance from } \theta_k \text{ to } \theta_{k+1} \text{ being } \sqrt{d}}$$

Can pick λ_k that maximizes $\bar{\ell}_T(\theta_k + \lambda_k g_k)$ (line search).

- "Steepest ascent" is only attained in a neighborhood of θ_k. Usually converges more slowly than BHHH.
- For minimization problems, this is called the gradient descent.

Stochastic Gradient Descent

- Historically gradient descent is not popular in nonlinear or nonconvex optimization problems because it gets stuck at local minima.
- For deep learning, computing the gradient can be very demanding. One way to be more efficient is the stochastic gradient descent (SGD).
- Instead of evaluating the gradient of full sample, we subsample $m \ll N$ observations with replacement and compute

$$\theta_{k+1} = \theta_k - \lambda_k g_k^\star.$$

  where $g_k^\star$ denotes the gradient of the subsample evaluated at θ_k.
- Practitioners prefer m small ($m = 1$): cheaper to compute and avoids overfitting (Goodfellow et al., 2016).

Outline

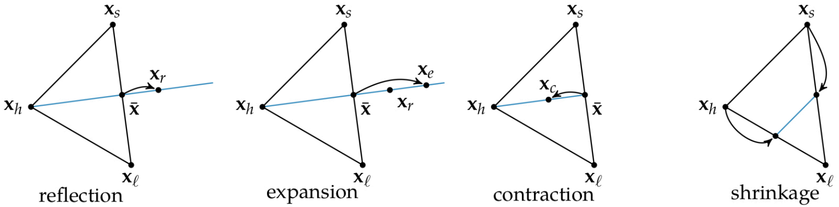
Comparison Methods

- Gradient-based methods are susceptible to converge at a local maximum. Can use a variety of starting values to investigate the issue.
- An alternative is comparison-based methods: compute objective function at several points and pick the one yielding the optimum value.
- Comparison methods better behave with non-smooth objective functions. Stochastic comparison methods are more likely to find global optimum (in theory).

Comparison Methods

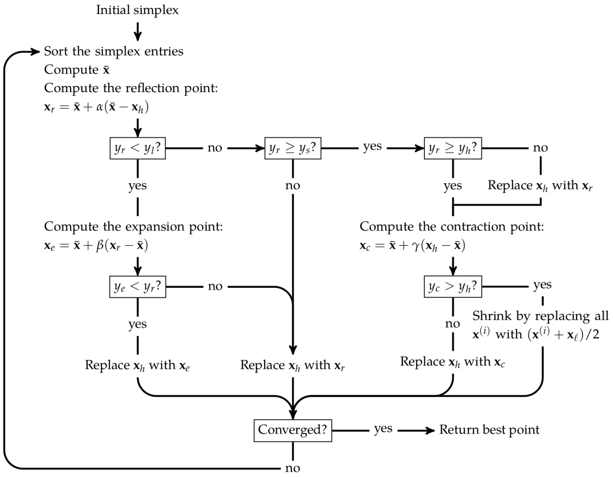
This is fminsearch in Matlab. Consider minimizing a generic criterion function $y = f(x)$ with $x \in \mathbb{R}^K$.

1. Choose initial simplex $\{x_1, x_2, \ldots, x_{n+1}\}$. Think of it as an n-dimensional version of a triangle.
2. Sort simplex vertices in descending order: $f(x_h) > f(x_s) > \cdots > f(x_l)$ (h: highest; s: second highest; l: lowest).
3. Modify the simplex in each step using one of the simplex operations.



Comparison Methods

Nelder-Mead Algorithm


Initial simplex
Sort the simplex entries
Compute $\bar{\mathbf{x}}$
Compute the reflection point:
$\mathbf{x}_r = \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x}_h)$
$y_r < y_l$?
no
$y_r \geq y_s$?
yes
$y_r \geq y_h$?
no
yes
no
yes
Replace $\mathbf{x}_h$ with $\mathbf{x}_r$
Compute the expansion point:
$\mathbf{x}_e = \bar{\mathbf{x}} + \beta(\mathbf{x}_r - \bar{\mathbf{x}})$
Compute the contraction point:
$\mathbf{x}_c = \bar{\mathbf{x}} + \gamma(\mathbf{x}_h - \bar{\mathbf{x}})$
$y_e < y_r$?
no
$y_c > y_h$?
yes
yes
no
Shrink by replacing all $\mathbf{x}^{(i)}$ with $(\mathbf{x}^{(i)} + \mathbf{x}_\ell)/2$
Replace $\mathbf{x}_h$ with $\mathbf{x}_e$
Replace $\mathbf{x}_h$ with $\mathbf{x}_r$
Replace $\mathbf{x}_h$ with $\mathbf{x}_c$
Converged?
yes
Return best point
no

Bibliography

Berndt, E. R., Hall, B. H., Hall, R. E., and Hausman, J. A. (1974), “Estimation and inference in nonlinear structural models,” in *Annals of Economic and Social Measurement, Volume 3, number 4*, NBER, pp. 653–665.

Goodfellow, I., Bengio, Y., and Courville, A. (2016), *Deep Learning*, MIT Press, `http://www.deeplearningbook.org`.

Kochenderfer, M. J. and Wheeler, T. A. (2019), *Algorithms for optimization*, Mit Press.

Train, K. E. (2009), *Discrete Choice Methods with Simulation*, Cambridge University Press, 2nd ed.