EC708 Discussion 8
Numerical Optimization

Yan Liu[1]

Department of Economics
Boston University

March 26, 2021

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

Outline

Optimization Problem: MLE

Consider maximizing the (average) log-likelihood function

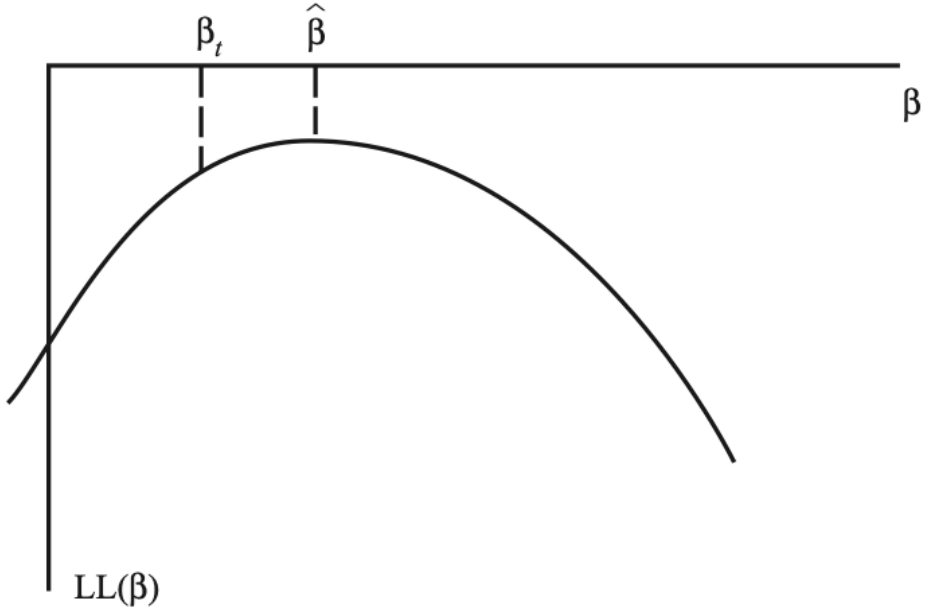
$$LL(\beta) = \frac{1}{N} \sum_{n=1}^{N} \ln P_n(\beta),$$

- $P_n(\beta)$: probability of the observed outcome for decision maker n
- N: sample size
- β: $K \times 1$ vector of parameters.

Goal: find $\hat{\beta} = \arg\max_{\beta} LL(\beta)$.

- To utilize minimization packages in practice, we usually work with $-LL(\beta)$.

Graphical Illustration of MLE



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

1. Specify a starting value β_0. Decide in which direction and how far to climb.
2. Each step t moves to a new value β_{t+1} at which $LL(\beta)$ is higher than at the current value β_t.
3. Keep climbing until no further increase can be found.

Gradient and Hessian

Gradient in step t:

$$g_t = \left(\frac{\partial LL(\beta)}{\partial \beta} \right)_{\beta_t}.$$

Hessian in step t:

$$H_t = \left(\frac{\partial g_t}{\partial \beta'} \right)_{\beta_t} = \left(\frac{\partial^2 LL(\beta)}{\partial \beta \partial \beta'} \right)_{\beta_t}.$$

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

Numerical Evaluation of Derivatives

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

$$\frac{\partial f(\beta)}{\partial \beta_j} = \lim_{h \to 0} \frac{f(\beta_1, \ldots, \beta_j + h, \ldots, \beta_K) - f(\beta_1, \ldots, \beta_j, \ldots, \beta_K)}{h}.$$

Numerically, we can approximate it by calculating

$$f_j(\beta) = \frac{f(\beta + hd_j) - f(\beta)}{h}, \quad j = 1, \ldots, K$$

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

$$f_j(\beta) = \frac{f(\beta + hd_j) - f(\beta - hd_j)}{2h}.$$

Newton-Raphson Method: Algorithm

Take a second-order Taylor's approximation of $LL(\beta_{t+1})$ around $LL(\beta_t)$:

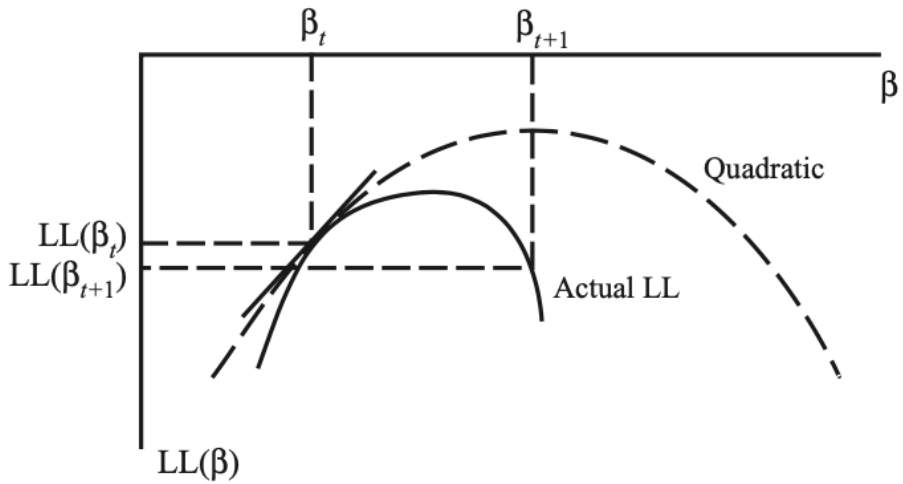
$$LL(\beta_{t+1}) = LL(\beta_t) + (\beta_{t+1} - \beta_t)' g_t + \frac{1}{2}(\beta_{t+1} - \beta_t)' H_t (\beta_{t+1} - \beta_t).$$

Find β_{t+1} that maximizes this approximation:

$$g_t + H_t(\beta_{t+1} - \beta_t) = 0 \;\Rightarrow\; \beta_{t+1} = \beta_t - \underbrace{H_t^{-1}}_{\text{step size}} \cdot \underbrace{g_t}_{\text{direction}}.$$

- Iterate until convergence, which can be defined in many ways:
  - $LL(\beta_{t+1})$ close to $LL(\beta_t)$
  - β_{t+1} close to β_t
  - g_{t+1} close to g_t
- If $LL(\beta)$ were exactly quadratic, then Newton-Raphson reaches the maximum in one step from any starting value.

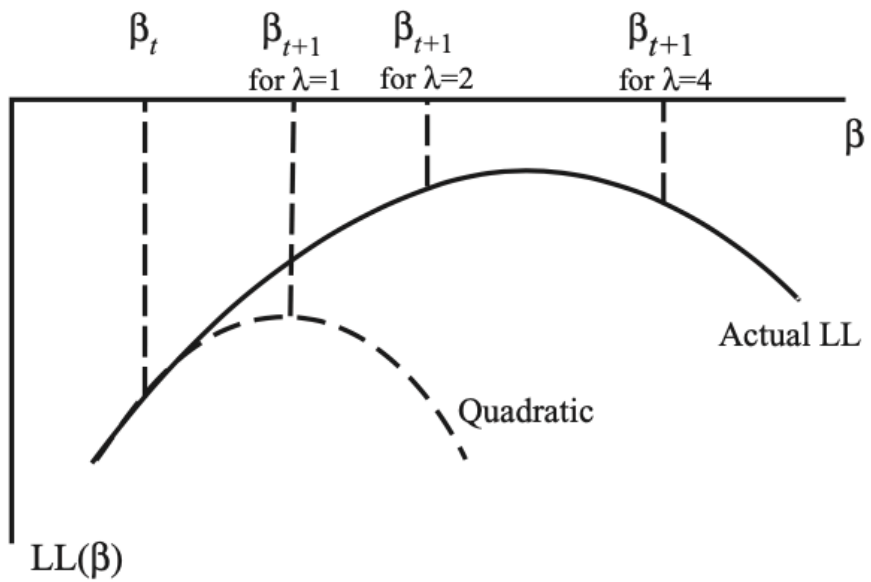
Newton-Raphson Method: Step Size



- It is possible for Newton-Raphson to step past the maximum and move to a lower $LL(\beta)$.
- To ensure each step provides an increase in $LL(\beta)$, we introduce a scalar step size λ_t:

$$\beta_{t+1} = \beta_t - \lambda_t H_t^{-1} g_t.$$

Newton-Raphson Method: Step Size



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

- If $LL(\beta_{t+1}) < LL(\beta_t)$, then set $\lambda_t = \frac{1}{2}$ and try again. Continue halving λ_t until a λ_t is found for which $LL(\beta_{t+1}) > LL(\beta_t)$.
  - A tiny λ_t is a signal that a different iteration procedure is needed.
- If $LL(\beta_{t+1}) > LL(\beta_t)$, then try $\lambda_t = 2$. Double λ_t as long as doing so further raises $LL(\beta_{t+1})$.
  - Raising λ_t 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_t^{-1} directly, use

$$(H_t + \mu_t I_K)^{-1}$$

where $\mu_t < 0$ guarantees negative definiteness.

Gauss-Newton Method

Consider the following general nonlinear model:

$$y_n = f(x_n; \beta) + u_n, \quad n = 1, \ldots, N$$

- $u_n \sim$ i.i.d.$N(0, \sigma^2)$. Assume σ^2 is known here.
- x_n: $M \times 1$ exogenous regressors.
- β: $K \times 1$ vector of parameters.
- $f(\cdot)$: some function satisfying some regularity conditions.

Gauss-Newton Method

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

$$LL(\beta) = -\frac{N}{2}\ln 2\pi - \frac{N}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}S(\beta)$$

where $S(\beta) = \sum_{n=1}^{N}[y_n - f(x_n;\beta)]^2 = \sum_{n=1}^{N} u_n^2$. It suffices to work with $S(\beta)$. Its first and second derivatives w.r.t. β are

$$g(\beta) = 2\sum_{n=1}^{N}\frac{\partial u_n}{\partial \beta}u_n, \quad H(\beta) = 2\sum_{n=1}^{N}\left[\frac{\partial u_n}{\partial \beta}\frac{\partial u_n}{\partial \beta'} + \frac{\partial^2 u_n}{\partial \beta \partial \beta'}u_n\right].$$

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

$$\beta_{t+1} = \beta_t - \left[\sum_{n=1}^{N}\frac{\partial u_n}{\partial \beta}\frac{\partial u_n}{\partial \beta'}\right]^{-1}\Bigg|_{\beta=\beta_t}\sum_{n=1}^{N}\frac{\partial u_n}{\partial \beta}u_n\Bigg|_{\beta=\beta_t}.$$

Gauss-Newton Method

Remarks:

- This method doesn't compute Hessian.
- Has an OLS interpretation. Let $z_n = -\partial u_n / \partial \beta$, then

$$\beta_{t+1} = \beta_t - \left(\sum_{n=1}^{N} z_n z_n' \right)^{-1} \Bigg|_{\beta=\beta_t} \sum_{n=1}^{N} z_n u_n \Bigg|_{\beta=\beta_t}.$$

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

$$\beta_{t+1} = \beta_t - \left(\sum_{n=1}^{N} z_n z_n' + \mu_t I_K \right)^{-1} \Bigg|_{\beta=\beta_t} \sum_{n=1}^{N} z_n u_n \Bigg|_{\beta=\beta_t}.$$

Quasi-Newton Methods

- Newton-Raphson method does not utilize the fact that the objective function is the sum of log likelihoods.
- Quasi-Newton methods do so by using the approximate Hessian instead of the actual one.

Quasi-Newton Methods: BHHH

Berndt, Hall, Hall, and Hausman (1974) propose to use scores to approximate Hessian.

- Score of observation n:

$$s_n(\beta_t) = \frac{\partial \ln P_n(\beta)}{\partial \beta}\Big|_{\beta=\beta_t}.$$

- Gradient is the average score:

$$g_t = \frac{1}{N}\sum_{n=1}^{N} s_n(\beta_t).$$

Quasi-Newton Methods: BHHH

- Outer product of observation n's score is the $K \times K$ matrix

$$s_n(\beta_t)s_n(\beta_t)' = \begin{pmatrix} s_n^1 s_n^1 & s_n^1 s_n^2 & \cdots & s_n^1 s_n^K \\ s_n^2 s_n^1 & s_n^2 s_n^2 & \cdots & s_n^2 s_n^K \\ \vdots & \vdots & & \vdots \\ s_n^K s_n^1 & s_n^K s_n^2 & \cdots & s_n^K s_n^K \end{pmatrix}$$

  where s_n^j is the j-th element of $s_n(\beta_t)$.

- Average outer product of scores:

$$B_t = \frac{1}{N} \sum_{n=1}^{N} s_n(\beta_t)s_n(\beta_t)'.$$

BHHH procedure:

$$\beta_{t+1} = \beta_t + \lambda_t B_t^{-1} g_t.$$

Quasi-Newton Methods: BHHH

Why does BHHH work?

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

Drawbacks: BHHH can give small steps when far from the maximum

- either because B_t is not a good approximation to $-H_t$,
- or because $LL(\beta)$ is highly nonquadratic since BHHH is an approximation to NR.

Quasi-Newton Methods: BFGS and DFP

- BHHH uses only information at β_t to determine each step.
- In contrast, BFGS and DFP use information at several points to obtain a sense of the curvature of the log-likelihood function.
- BFGS is the algorithm behind Matlab's fminunc.

BFGS and DFP updates have the form

$$\beta_{t+1} = \beta_t + \lambda_t Q_t g_t.$$

Quasi-Newton Methods: BFGS and DFP

DFP sets Q_0 to identity matrix and updates Q_{t+1} from Q_t using

$$Q_{t+1} = Q_t + \frac{Q_t \gamma_t \gamma_t' Q_t}{\gamma_t' Q_t \gamma_t} + \frac{\delta_t \delta_t'}{\delta_t' \gamma_t}$$

where $\delta_t = \beta_{t+1} - \beta_t$ and $\gamma_t = g_{t+1} - g_t$. BFGS further subtracts $v_t d_t d_t'$ from the DFP update, where

$$v_t = \gamma_t' Q_t \gamma_t, \quad d_t = \frac{\delta_t}{\delta_t' \gamma_t} - \frac{Q_t \gamma_t}{\gamma_t' Q_t \gamma_t}.$$

There is some evidence that BFGS is more efficient than DFP.

Steepest Ascent

The greatest possible increase in $LL(\beta)$ for the (small enough) distance between β_t and β_{t+1} is provided by

$$\beta_{t+1} = \beta_t + \lambda_t g_t.$$

Motivated by the Lagrangian:

$$L = \underbrace{LL(\beta_t) + (\beta_{t+1} - \beta_t) g_t}_{\text{1st-order Taylor expansion of } LL(\beta_{t+1})} - \frac{1}{2\lambda_t} \underbrace{[(\beta_{t+1} - \beta_t)'(\beta_{t+1} - \beta_t) - k]}_{\text{distance from } \beta_t \text{ to } \beta_{t+1} \text{ being } \sqrt{k}}$$

Can pick λ_t that maximizes $LL(\beta_t + \lambda_t g_t)$ (line search).

- "Steepest ascent" is only attained in a neighborhood of β_t. 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, local minima are close to global minima in terms of prediction, so not problematic.
- 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

$$\beta_{t+1} = \beta_t - \lambda_t g_t^\star.$$

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

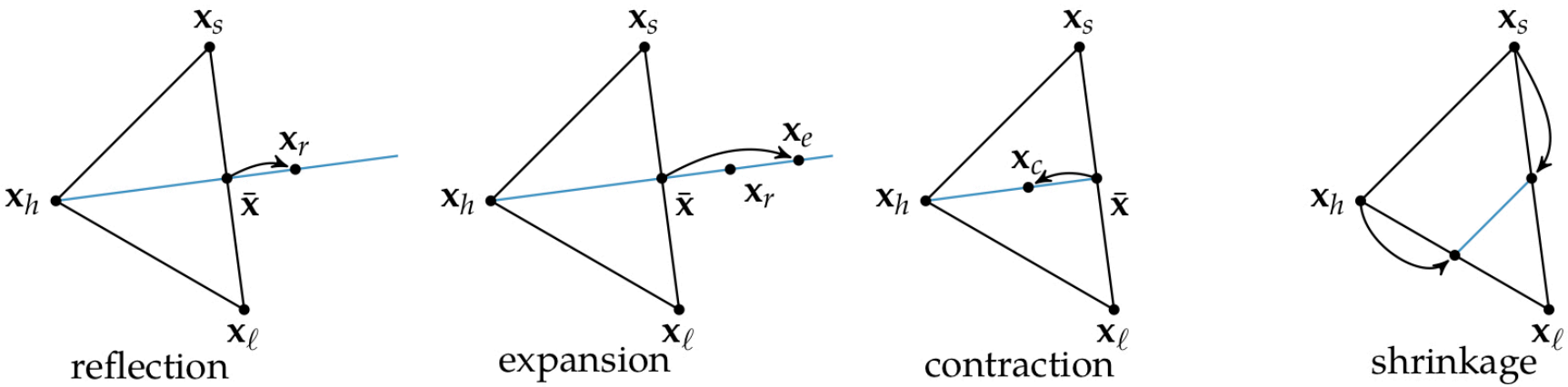
Comparison Methods

- 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).

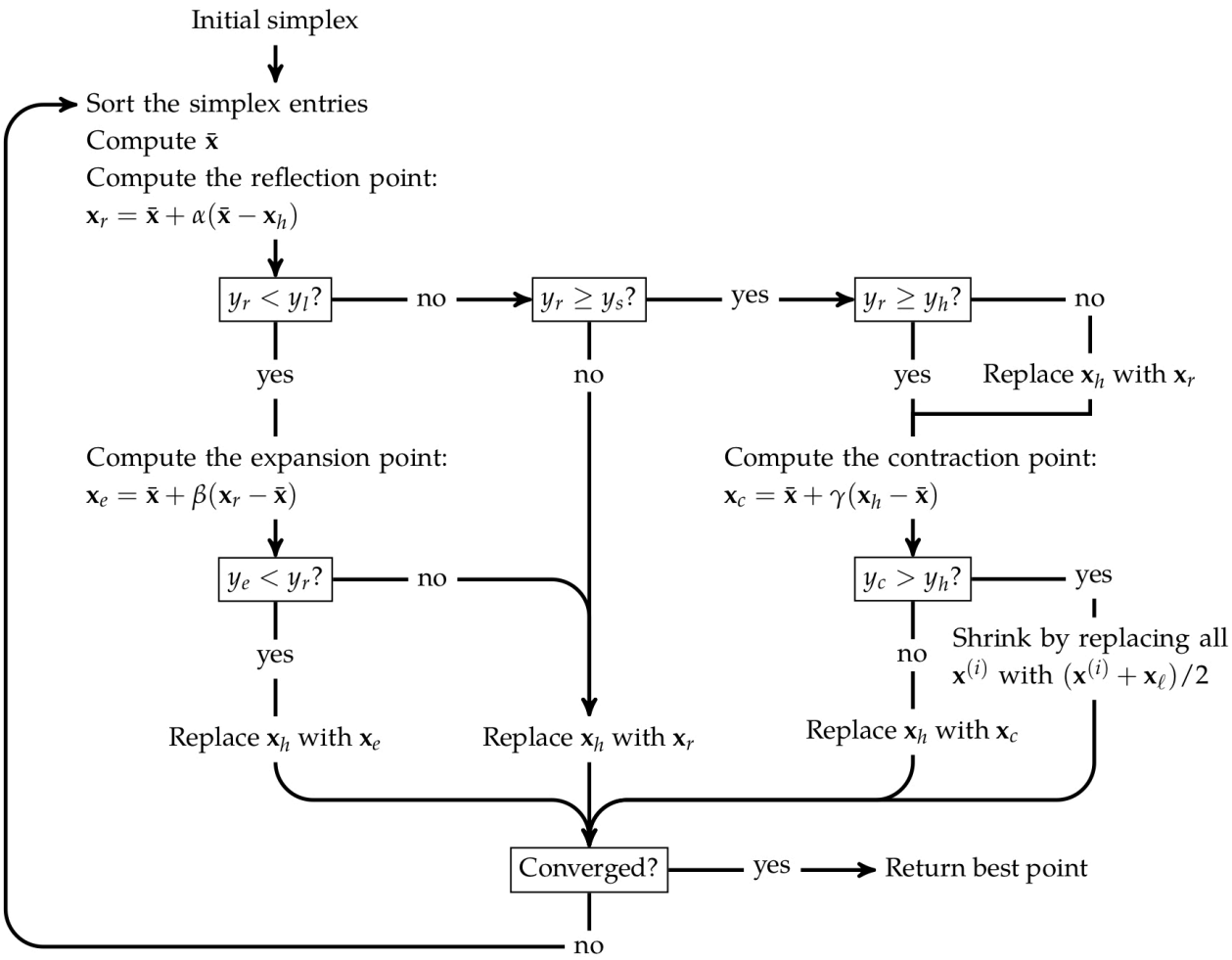
Nelder-Mead Algorithm

This is fminsearch in Matlab. Consider minimizing a generic criterion function $f(x)$ 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).



Nelder-Mead Algorithm



Intuition: Nelder-Mead starts with a simplex and modifies it at each iteration using one of the simplex operations.

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.